SIEMENS

SIMATIC

SIMIT Simulation Platform (V10.3)

Operating Manual

Preface

Basics of SIMIT 1

Couplings 2

Simulation model 3

Automatic model creation 4

Diagnostics & visualization 5

Scripts 6

Component type editor 7

Libraries 8

Licensing 9

Menus and dialog boxes 10

06/2021
A5E44876249-AC

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

indicates that minor personal injury can result if proper precautions are not taken.

NOTICE

indicates that property damage can result if proper precautions are not taken.

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
Digital Industries
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E44876249-AC
Ⓟ 08/2021 Subject to change

Copyright © Siemens AG 2021.
All rights reserved

Preface

Security disclaimer for simulation products

Siemens' simulation products allow you to simulate and/or optimize the planning and operation of a plant/machine. The simulation and optimization results merely represent non-binding recommendations and are dependent on the completeness and correctness of the input data. The user must therefore check the input data and results for plausibility after each simulation/ optimization.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit Siemens Industrial Security (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under Siemens Industrial Security RSS Feed (https://www.siemens.com/industrialsecurity).

Security hardening

With the installation of SIMIT SP or SIMIT VC, settings are made in the firewall which enable communication with some processes associated with SIMIT via network protocols.

In business-critical or safety-relevant applications, operation of SIMIT in a network separate from the plant is recommended. At a minimum, the firewall settings should be checked and made more restrictive, if necessary.

When operating SIMIT inside a virtual environment, customers are responsible for security hardening of the virtualization infrastructure, especially virtual machine images as well as virtual network configurations including access permissions.

What's new?

Please refer to the ReadMe file, which you will find in the root directory of the installation data storage medium, for information on the new functions offered by this version of SIMIT. After the installation, you will also find this file via the Windows start menu in the folder "SIMIT | Documentation".

Virus scanners

The following virus scanners are checked for compatibility with SIMIT:

- Trend Micro V12.0 SP1
- Symantec Endpoint Protection V14.2
- McAfee Endpoint Security V10.5
- Microsoft Defender

Note

When using McAfee Endpoint Security with SIMIT VC, on-access scanning must be disabled on all PCs for the following directories (including subfolders):

- Storage directory of SIMIT projects (user-specific)
- C:\ProgramData\Siemens\Automation\SIMIT VC\9.0

Refer to the manual for your virus scanner for information on configurable options.

Always agree on the security settings of your systems with your system administrators and security administrators.

You may find information on the currently approved antivirus software versions on the SIMIT support pages. To go to the support pages, select "Help" > "SIMIT resources" in the menu bar.

Table of contents

Preface **3**

1 Basics of SIMIT **19**

1.1 Operating principle 19

1.2 DEMO mode 19

1.3 Starting SIMIT 21

1.4 The SIMIT graphical user interface 22

1.5 Creating a simulation 24
1.5.1 Structure of a simulation 24
1.5.2 Visualizing a simulation 26
1.5.3 Visualizing coupling signals 29

1.6 Simulation 32
1.6.1 The states of a simulation 32
1.6.2 Simulation operator control and monitoring 34
1.6.2.1 Actions during ongoing simulation 34
1.6.2.2 Display of the simulation load 35
1.6.2.3 Changes in a simulation project while a simulation is running 37
1.6.3 Settings for simulation 39
1.6.3.1 Operating modes 39
1.6.3.2 Time slices 43
1.6.3.3 Speeding up and slowing simulation 44
1.6.3.4 Adjustable timeout times 45

1.7 Distributed simulation 46
1.7.1 Method of operation of the distributed simulation 46
1.7.2 Configuration of a distributed simulation 47
1.7.3 Data exchange in a distributed simulation 49
1.7.4 Setup firewall 50
1.7.5 Create screen link "SIMIT Client" 51
1.7.6 Start distributed simulation 52
1.7.7 Changing distributed simulation on a client PC 53

2 Couplings **55**

2.1 Coupling concept 55
2.1.1 Couplings in the SIMIT architecture 55
2.1.2 Couplings to SIMATIC PLCs 57

2.2 Coupling editor 59

2.3 Activation and deactivation of couplings 60
2.3.1 Activation and deactivation of couplings 60
2.3.2 Activating couplings 61
2.3.3 Deactivating couplings 61
2.3.4 Revoking coupling assignments 62

2.4 HWCNExport 62

2.4.1 HWCNExport introduction ... 62
2.4.2 Creating a HWCNExport file from STEP 7 projects ... 63
2.4.3 Creating an HWCNExport file from TIA projects ... 64

2.5 SIMIT Unit ... 65
2.5.1 Operating principle of the SIMIT Unit coupling ... 65
2.5.2 Supported PROFIBUS DP configurations ... 67
2.5.3 Supported PROFINET IO configurations ... 81
2.5.4 Slave/I-device ... 88
2.5.5 Redundant and fail-safe systems ... 89
2.5.5.1 Redundant configurations (H systems) ... 89
2.5.5.2 Fail-safe configurations (F systems) ... 93
2.5.5.3 Redundant and fail-safe configurations (H/F systems) ... 93
2.5.6 Setting up and managing SIMIT Unit ... 94
2.5.6.1 Configuring a SIMIT Unit ... 94
2.5.6.2 Importing device description file to SIMIT ... 95
2.5.6.3 Updating the firmware of the SIMIT Unit ... 95
2.5.6.4 Resetting SIMIT Unit ... 96
2.5.7 Configuring a coupling of the "SIMIT Unit" type ... 97
2.5.7.1 Creating a coupling of the "SIMIT Unit" type ... 97
2.5.7.2 Creating a station ... 98
2.5.7.3 Copying system data blocks ... 98
2.5.7.4 Importing hardware configuration to station ... 99
2.5.7.5 Shared devices ... 100
2.5.7.6 Modules that cannot be uniquely identified ... 101
2.5.7.7 DP/DP coupler ... 104
2.5.7.8 Configuring the station ... 107
2.5.7.9 Simulation of alarms ... 108
2.5.7.10 Device status ... 110
2.5.7.11 Disabling a line, device or module ... 111
2.5.7.12 Loading configuration to station ... 112
2.5.7.13 Properties of the SIMIT Unit coupling ... 113
2.5.7.14 XML import interface for hardware configuration of third-party systems ... 116

2.6 Virtual Controller ... 122
2.6.1 Operating principle of Virtual Controllers ... 122
2.6.1.1 Introduction ... 122
2.6.1.2 Requirements for a simulation network ... 126
2.6.1.3 Functions ... 127
2.6.1.4 Basics of Virtual Controller communication ... 128
2.6.1.5 Supported system functions ... 130
2.6.1.6 Supported S7 blocks ... 131
2.6.1.7 Supported services ... 132
2.6.1.8 Handling sync errors ... 135
2.6.1.9 Sequential control ... 136
2.6.1.10 Data record communication ... 137
2.6.1.11 Substitute mechanism for data exchange ... 138
2.6.1.12 Notes on use of a virtual controller ... 138
2.6.2 Configuring the Virtual Controllers ... 139
2.6.2.1 Configuring virtual controllers ... 139
2.6.2.2 Creating a Virtual Controller ... 141
2.6.2.3 Importing a STEP 7/PCS 7 project ... 142
2.6.2.4 Configuring virtual controllers ... 143

2.6.2.5 Distribution editor (Virtual Controller) ... 144
2.6.2.6 Restarting the virtual controller ... 148
2.6.2.7 Configuring an additional PC ... 149
2.6.2.8 Virtual controller during simulation ... 150
2.6.3 Editing signals ... 152
2.6.3.1 Properties of the Virtual Controller coupling ... 152
2.6.3.2 Importing DB data from STL sources ... 155

2.7 PLCSIM Advanced coupling ... 157
2.7.1 Operating principle of the PLCSIM Advanced coupling ... 157
2.7.2 Creating a coupling of the "PLCSIM Advanced" type ... 157
2.7.3 Importing hardware configuration to station ... 158
2.7.4 Configuring a PLCSIM Advanced coupling station ... 159
2.7.5 Distribution editor (PLCSIM Advanced) ... 160

2.8 PLCSIM coupling ... 163
2.8.1 How the PLCSIM coupling works ... 163
2.8.2 Configuring the PLCSIM coupling ... 163
2.8.2.1 Creating a PLCSIM coupling ... 163
2.8.2.2 Configuring I/O signals in the PLCSIM coupling ... 163
2.8.2.3 Properties of the PLCSIM coupling ... 164

2.9 OPC coupling ... 165
2.9.1 How the OPC coupling works ... 165
2.9.2 Signal transfer between OPC server and OPC client ... 166
2.9.3 Mapping of signals and data types of an OPC server ... 168
2.9.4 Status of OPC servers ... 169
2.9.5 Configuring an OPC DA client/server coupling ... 171
2.9.5.1 Principle of operation of the SIMIT OPC DA server ... 171
2.9.5.2 Configuring a SIMIT OPC DA server coupling ... 172
2.9.5.3 Configuring an OPC DA client coupling ... 173
2.9.5.4 Properties of the SIMIT OPC DA server coupling ... 174
2.9.5.5 Properties of the OPC DA client coupling ... 175
2.9.5.6 Notes on DCOM configuration ... 176
2.9.6 Configuring the OPC UA client/server coupling ... 177
2.9.6.1 Operating principle of the SIMIT OPC UA server ... 177
2.9.6.2 Setting up the firewall for OPC UA server ... 179
2.9.6.3 Configuring a SIMIT OPC UA server coupling ... 180
2.9.6.4 Properties of the OPC UA server coupling ... 181
2.9.6.5 Principle of operation of the OPC UA client coupling ... 181
2.9.6.6 Configuring an OPC UA client coupling ... 182
2.9.6.7 Error messages upon connection to the OPC UA server ... 183
2.9.6.8 Properties of the OPC UA client coupling ... 184
2.9.6.9 Operating principle of the SIMIT gPROMS coupling ... 185
2.9.6.10 Enabling a gPROMS co-simulation ... 187
2.9.6.11 Configuring a SIMIT gPROMS coupling ... 188
2.9.6.12 Properties of the gPROMS coupling ... 188
2.9.6.13 Additional properties of the signals of a gPROMS coupling ... 189
2.9.6.14 Certificates ... 189

2.10 Shared memory coupling ... 191
2.10.1 How the SHM coupling works ... 191
2.10.1.1 Accessing the memory area ... 191
2.10.1.2 Structure of the memory area ... 192

2.10.1.3 Creating the memory area ... 195
2.10.2 Configuring the SHM coupling ... 196
2.10.2.1 Creating an SHM coupling ... 196
2.10.2.2 Configuring the signals in the SHM coupling ... 196
2.10.2.3 Signal properties in the SHM coupling ... 197
2.10.2.4 Properties of the SHM coupling ... 198
2.10.2.5 Importing and exporting signals ... 199
2.11 Plant Simulation ... 200
2.12 PRODAVE coupling ... 200
2.12.1 How the PRODAVE coupling works ... 200
2.12.2 Configuring the PRODAVE coupling ... 201
2.12.2.1 Creating a PRODAVE coupling ... 201
2.12.2.2 Editing signals in the PRODAVE coupling ... 201
2.12.2.3 Properties of the PRODAVE coupling ... 201
2.12.2.4 Importing the signal properties ... 202
2.12.2.5 Exporting the signal properties ... 202
2.13 Mechatronics Concept Designer coupling ... 203
2.13.1 How the Mechatronics Concept Designer coupling works ... 203
2.13.2 Usage scenarios of the Mechatronics Concept Designer coupling ... 204
2.13.3 Configuring Mechatronics Concept Designer coupling ... 206
2.13.3.1 Creating the Mechatronics Concept Designer coupling ... 206
2.13.3.2 Importing signals ... 206
2.13.3.3 Editing signal properties in the Mechatronics Concept Designer coupling ... 207
2.13.3.4 Configuring the isochronous Mechatronics Concept Designer coupling ... 208
2.13.3.5 Properties of the Mechatronics Concept Designer coupling ... 208
2.14 Editing the signals of a coupling ... 210
2.14.1 Signal basics ... 210
2.14.1.1 Coupling data direction ... 210
2.14.1.2 Meaning of the coupling name ... 210
2.14.1.3 Sorting and filtering of signals in the coupling editor ... 211
2.14.1.4 Addressing signals ... 212
2.14.1.5 Fixing signals in the coupling editor ... 212
2.14.1.6 Mapping of SIMATIC data types in SIMIT ... 215
2.14.1.7 Access to a data record or memory area ... 216
2.14.1.8 Converting the data width of signals ... 217
2.14.1.9 Scaling analog signals ... 218
2.14.2 Importing and exporting signals ... 221
2.14.2.1 File formats for signals ... 221
2.14.2.2 Symbol table ... 222
2.14.2.3 Variable table ... 223
2.14.2.4 Signal table ... 223
2.14.2.5 INI format of the OPC couplings ... 224
2.14.2.6 Spreadsheet format (xlsx) of the gPROMS coupling ... 225
2.14.2.7 JSON format of the gPROMS coupling ... 226
2.14.2.8 Importing signal properties ... 227
2.14.2.9 Importing PLC variable lists ... 228
2.14.2.10 Exporting signal properties ... 228
2.14.3 Using I/O signals ... 230
2.14.3.1 I/O signals for connection on charts ... 230
2.14.3.2 I/O signals for animations ... 231

2.14.3.3 I/O signals in controls on charts ... 232
2.14.3.4 I/O signals in trends ... 232
2.14.3.5 Multiple use of I/O connectors ... 233
2.14.4 Interconnecting signals ... 233
2.14.5 Splitting a signal ... 234
2.14.6 Combining signals ... 235
2.14.7 Preassigning signals ... 236
2.14.8 Fixing a signal ... 237
2.14.9 Addressing a signal symbolically ... 237
2.14.10 Scaling signals ... 238
2.14.11 Transferring scaling to another signal ... 239
2.14.12 Limiting a signal ... 240
2.14.13 Reading signals from an OPC server ... 241
2.14.14 Configuring a signal "With readback capability" ... 242

2.15 TableReader coupling ... 244
2.15.1 Functionality of the TableReader coupling ... 244
2.15.1.1 Functionality of the "TableReader" coupling ... 244
2.15.1.2 Structure of a TableReader table ... 244
2.15.2 Configuring the TableReader coupling ... 244
2.15.2.1 Creating a TableReader coupling ... 244
2.15.2.2 Importing a table ... 244

3 Simulation model ... 247

3.1 Project manager ... 247
3.1.1 View and functions of the Project Manager ... 247
3.1.2 Properties ... 248
3.1.3 Versioning ... 249
3.1.4 Write protection ... 251

3.2 Chart editor ... 252
3.2.1 Creating and editing charts ... 252
3.2.2 Visualizing graphics ... 254
3.2.3 Visualizing signals ... 257
3.2.4 Printing charts ... 258
3.2.5 Activation and deactivation of charts ... 258
3.2.5.1 Activation and deactivation of charts ... 258
3.2.5.2 Activating charts ... 259
3.2.5.3 Deactivating charts ... 260

3.3 Task cards ... 261
3.3.1 "Components" task card ... 261
3.3.2 "Controls" task card ... 263
3.3.3 "Macros" task card ... 265
3.3.4 "Graphic" task card ... 266
3.3.5 "Templates" task card ... 267
3.3.6 "Projects" task card ... 268
3.3.7 "Signals" task card ... 270

3.4 Macro component editor ... 271
3.4.1 Macro editor ... 271
3.4.2 Inserting dividing lines between macro component I/Os ... 274
3.4.3 Topological connectors of macro components ... 275
3.4.4 Default settings for macro component inputs ... 275

3.4.5 Defining parameters of macro components ... 276
3.4.6 Properties of macro components ... 276
3.4.7 Find and Replace in macro components ... 277
3.4.8 Using macro components ... 277

3.5 Migrating projects from previous versions of SIMIT ... 278

4 Automatic model creation ... 279

4.1 Templates ... 280
4.1.1 Find and replace in templates ... 282
4.1.2 Instantiating templates by entering replacements ... 282
4.1.3 Creating tables from a template ... 283
4.1.4 Defining a folder hierarchy for templates ... 284
4.1.5 Addressing a module via the I/O address ... 284
4.1.6 Indirect addressing ... 285
4.1.7 Opening basic templates in the editor ... 286

4.2 Instantiation of templates from files or the simulation model ... 286
4.2.1 Instantiating templates from files ... 286
4.2.2 Table import ... 287
4.2.3 IEA import ... 291
4.2.4 CMT import ... 293
4.2.4.1 Instantiating templates for interconnection between CMTs ... 297
4.2.5 Instantiating templates from the simulation model ... 298

4.3 Automated import ... 299
4.3.1 Import charts from ZIP or XML files ... 299
4.3.2 Syntax of the XML file ... 300
4.3.3 Examples of XML files ... 304
4.3.4 Structure of the control file "procedure.cfg" for import from ZIP file ... 304

4.4 Bulk engineering ... 305
4.4.1 Exporting data ... 305
4.4.2 Content of the export file ... 306
4.4.3 Importing data ... 307

5 Diagnostics & visualization ... 309

5.1 Trend and messaging editor ... 309
5.1.1 Functions of the Trend and Messaging Editor ... 309
5.1.2 Message system ... 309
5.1.2.1 Message classes ... 309
5.1.2.2 Message editor ... 310
5.1.2.3 Editing messages ... 310
5.1.2.4 Message – the message component ... 311
5.1.2.5 Component-specific messages ... 311
5.1.2.6 Messages in the status bar ... 312
5.1.2.7 Limitations of the message system ... 312
5.1.3 Archive ... 312
5.1.4 Trends ... 314
5.1.4.1 Signal with trends ... 314
5.1.4.2 Adding and configuring trends ... 314
5.1.4.3 Displaying trends ... 317
5.1.4.4 Trend editor ... 322

5.2 Find & replace ... 323

5.2.1 Find ... 324
5.2.1.1 Finding with the Find & replace editor ... 324
5.2.1.2 Searching using the properties of signals and connectors ... 326
5.2.1.3 Searching for fixed signals ... 326
5.2.2 Replace ... 326
5.2.2.1 Replacing with the Find & Replace editor ... 326
5.2.2.2 Refresh ... 327
5.3 Consistency check ... 328
5.4 Analysis functions ... 329
5.4.1 Basics of the analysis function ... 329
5.4.2 Generating analyses ... 331
6 Scripts ... 333
6.1 How scripts work ... 333
6.2 Handling of scripts ... 334
6.2.1 Creating a script ... 334
6.2.2 Executing a script ... 335
6.2.3 External scripts ... 336
6.3 Script syntax ... 337
6.3.1 Controlling the script ... 337
6.3.2 Composing scripts ... 338
6.3.3 Commenting scripts ... 339
6.3.4 Signals in scripts ... 339
6.3.5 Controlling the simulation ... 340
6.3.5.1 Initializing a simulation ... 341
6.3.5.2 Starting the simulation ... 341
6.3.5.3 Starting and waiting until an absolute time ... 341
6.3.5.4 Starting and waiting until a relative time ... 342
6.3.5.5 Starting and waiting for a certain number of time slices ... 342
6.3.5.6 Starting and waiting for an event ... 342
6.3.5.7 Stopping the simulation ... 342
6.3.5.8 Executing a single step ... 342
6.3.5.9 Saving a snapshot ... 343
6.3.5.10 Loading a snapshot ... 343
6.3.5.11 Resetting the simulation time ... 343
6.3.6 Logging ... 344
6.3.6.1 Opening and closing log files ... 344
6.3.6.2 Unformatted output ... 344
6.3.6.3 Formatted output ... 345
6.3.6.4 Outputting time and date ... 345
6.3.6.5 Outputting version information ... 346
6.3.6.6 The _printlog system function ... 346
6.3.7 Signal curves ... 347
6.3.7.1 Opening and closing a plot file ... 347
6.3.7.2 Specifying signals ... 348
6.3.7.3 Specifying a cycle ... 348
6.3.8 Setting signals ... 348
6.3.8.1 Setting individual values ... 349
6.3.8.2 Separating connected signals ... 349
6.3.8.3 Specifying signals ... 350

6.3.9 Conditional execution ... 350
6.3.10 Accessing the simulation time ... 352

7 Component type editor ... 353

7.1 User interface ... 353
7.1.1 Starting the CTE ... 353
7.1.2 Structure of the user interface ... 355
7.1.3 Menu bar and toolbar ... 357
7.1.4 Project tree ... 358
7.1.5 Keyboard shortcuts ... 358
7.1.6 The task cards ... 359
7.1.6.1 The "Signals" task card ... 359
7.1.6.2 The "Controls" task card ... 360
7.1.6.3 The "Components" task card ... 361
7.1.6.4 The "Enumeration Types" task card ... 362
7.1.6.5 The "Connection Types" task card ... 363

7.2 Principles of component types ... 365
7.2.1 The SIMIT type-instance concept ... 365
7.2.2 Properties of component types ... 366
7.2.3 General properties of component types ... 367
7.2.3.1 Overview ... 367
7.2.3.2 Administration properties ... 369
7.2.3.3 Protection of the component type ... 369
7.2.3.4 Special properties ... 370
7.2.3.5 Change comments ... 371

7.3 Connectors and parameters of component types ... 371
7.3.1 The connector editor ... 371
7.3.2 Special default setting for implicitly interconnectable inputs ... 374
7.3.3 Complex connection types ... 376
7.3.4 Parameters of component types ... 377

7.4 The behavior of a component type ... 379
7.4.1 Introduction ... 379
7.4.2 State variables ... 379
7.4.3 Initialization, cyclic calculation and functions ... 381
7.4.4 Data record handler ... 382
7.4.5 Topology ... 384

7.5 Visualization of component types ... 384
7.5.1 Introduction ... 384
7.5.2 The basic symbol ... 385
7.5.2.1 Editing the basic symbol ... 385
7.5.2.2 Editing graphics ... 385
7.5.2.3 Editing connectors ... 386
7.5.2.4 Editing properties ... 386
7.5.2.5 Scalability ... 388
7.5.3 The link symbol ... 389
7.5.4 The operating window ... 390

7.6 Documentation of component types ... 393

7.7 Syntax for the behavior description ... 393
7.7.1 Overview ... 393

7.7.2 Conversion of the behavior description to C# code ... 394
7.7.3 The instruction-oriented approach ... 394
7.7.3.1 Introduction ... 394
7.7.3.2 Functions ... 394
7.7.3.3 Blocks ... 395
7.7.3.4 Local variables ... 396
7.7.3.5 Fields ... 396
7.7.3.6 Constants ... 397
7.7.3.7 Loops ... 397
7.7.3.8 Conditional branches ... 398
7.7.3.9 System functions ... 399
7.7.3.10 Operators ... 400
7.7.3.11 Accessing state variables ... 402
7.7.4 Differential equation systems ... 402
7.7.4.1 Introduction ... 402
7.7.4.2 Notation for the differential ... 402
7.7.4.3 Corrections for the state variables ... 403
7.7.4.4 Accessing continuous state variables ... 403
7.7.5 The equation-oriented approach ... 403
7.7.5.1 Overview ... 403
7.7.5.2 Local variables ... 404
7.7.5.3 Constants ... 405
7.7.5.4 The calculation order ... 405
7.7.5.5 Operators ... 406
7.7.5.6 Conditional assignments ... 407
7.7.5.7 Enumeration types ... 407
7.7.5.8 Vectors ... 408
7.7.5.9 Function calls for mathematical standard functions ... 409
7.7.5.10 User-defined functions ... 410
7.7.5.11 Accessing discrete state variables ... 411
7.7.6 Internal variables and constants ... 411
7.7.7 The characteristic parameter type ... 412

7.8 Functional Mock-up Units (FMU) ... 412
7.8.1 Operating principle of the FMU integration ... 412
7.8.1.1 Conversion via component type editor / CTE ... 413
7.8.1.2 Conversion via SIMIT SP ... 414
7.8.1.3 Placing and interconnecting the FMU components ... 414
7.8.2 Supported FMU properties and compatibility ... 414
7.8.2.1 Necessary properties ... 414
7.8.2.2 Supported properties ... 415
7.8.2.3 Unsupported properties ... 415
7.8.3 Automatic compatibility adaptations ... 416
7.8.4 Extended functions and diagnostics ... 417
7.8.4.1 Reading of FMU states including derivation ... 417
7.8.4.2 Diagnostic capabilities of an FMU under error conditions ... 417

8 Libraries ... 419

8.1 Basic library ... 419
8.1.1 General ... 419
8.1.1.1 Introduction ... 419
8.1.1.2 Component symbols ... 419
8.1.1.3 Symbols for the controls ... 421

8.1.1.4 Structure of the simulation model ... 422
8.1.1.5 Component connectors ... 422
8.1.1.6 Connectors for controls ... 423
8.1.1.7 Connecting I/O ... 423
8.1.1.8 Setting inputs ... 426
8.1.1.9 Properties of components ... 426
8.1.1.10 Component error messages ... 433
8.1.1.11 Properties of controls ... 433
8.1.2 Connectors ... 435
8.1.2.1 Global connector ... 436
8.1.2.2 I/O connectors ... 437
8.1.2.3 Topological connector ... 437
8.1.2.4 Unit connector ... 437
8.1.3 Standard components ... 438
8.1.3.1 Differentiating standard components ... 438
8.1.3.2 Analog functions ... 438
8.1.3.3 Integer functions ... 461
8.1.3.4 Mathematical functions ... 468
8.1.3.5 Binary functions ... 473
8.1.3.6 Converting values ... 482
8.1.3.7 General components in the Misc directory ... 492
8.1.4 Drive components ... 501
8.1.4.1 Drives as component types ... 501
8.1.4.2 Valve drives ... 501
8.1.4.3 Pump drives and fan drives ... 504
8.1.4.4 Motor ... 506
8.1.4.5 PROFIdrive devices ... 512
8.1.4.6 SIMOCODE pro motor control devices ... 551
8.1.5 Communication components ... 570
8.1.5.1 Components for SIMATIC ... 570
8.1.5.2 Components for SINUMERIK ... 580
8.1.6 Controls ... 584
8.1.6.1 Controls for displaying signal values ... 584
8.1.6.2 Controls for entering signal values ... 590
8.1.6.3 Miscellaneous controls ... 600
8.1.6.4 3D Viewer control ... 602

8.2 CHEM-BASIC and FLOWNET libraries ... 615
8.2.1 Introduction ... 615
8.2.2 Topological connector ... 616
8.2.3 Geodetic height ... 618
8.2.4 Flownets ... 620
8.2.4.1 Flownet basics ... 620
8.2.4.2 Variables used in flownets ... 623
8.2.4.3 Modeling of flownet branches ... 623
8.2.4.4 Modeling of flownet nodes ... 624
8.2.4.5 Heat exchange with the environment ... 627
8.2.4.6 Parameter assignment of flownets ... 627
8.2.4.7 Coupling simulation model with actuator/sensor level ... 631
8.2.5 Components of the CHEM-BASIC library ... 634
8.2.5.1 Burner ... 634
8.2.5.2 Fittings ... 635
8.2.5.3 Graphics ... 642

8.2.5.4 Heatexchanger ... 643
8.2.5.5 Measurements ... 657
8.2.5.6 Mixing apparatuses ... 667
8.2.5.7 Pumps ... 670
8.2.5.8 Reactors ... 687
8.2.5.9 Separators ... 695
8.2.5.10 System ... 746
8.2.5.11 Tanks ... 752
8.2.5.12 Valves ... 780
8.2.6 Components in FLOWNET library ... 792
8.2.6.1 General components ... 792
8.2.6.2 Measuring components ... 804
8.2.6.3 Component types for "water/steam" medium ... 810
8.2.6.4 Component types for liquid medium ... 827
8.2.6.5 Component types for gas medium ... 839
8.2.7 Creating your own component types for flownets ... 852
8.2.7.1 Topological properties ... 852
8.2.7.2 Connection to the solution procedure ... 855
8.2.7.3 Constants and functions ... 862
8.2.7.4 Initializing FLOWNET simulations ... 867

8.3 CONTEC library ... 867
8.3.1 Introduction ... 867
8.3.2 Material handling simulation ... 868
8.3.2.1 Principles of conveyor technology simulation ... 870
8.3.2.2 Modeling of the objects ... 871
8.3.2.3 Modeling the conveyor system network ... 873
8.3.2.4 Special features of conveyor technology simulation ... 876
8.3.2.5 Scalability ... 878
8.3.2.6 Generating the simulation of drives and sensors ... 881
8.3.3 Components of the CONTEC library ... 886
8.3.3.1 Topological connector in the CONTEC library ... 886
8.3.3.2 Component types for conveyor systems with vehicles ... 888
8.3.3.3 Component types for non-vehicular conveyor systems ... 904
8.3.3.4 Component types for simulating objects ... 938
8.3.4 Creating custom component types for material handling simulation ... 944
8.3.4.1 Topological properties ... 945
8.3.4.2 Connection to the solution procedure ... 948
8.3.4.3 System functions ... 956
8.3.4.4 System variables ... 965

9 Licensing ... 967

9.1 License model ... 967

9.2 Setting up WIBU license server ... 968

9.3 Setting up access permission to the license server ... 969

9.4 Setting up a server search list on the client ... 970

10 Menus and dialog boxes ... 971

10.1 Menus ... 971
10.1.1 Portal view > Start ... 971
10.1.2 Portal view > Couplings ... 972

10.1.3 Portal view > Simulation model ... 973
10.1.4 Portal view > Automatic model creation ... 973
10.1.5 Portal view > Diagnostics & visualization ... 974
10.1.6 Project > New project 974
10.1.7 Project > Open... ... 975
10.1.8 Project > Close ... 975
10.1.9 Project > Save all ... 976
10.1.10 Project > Save as... ... 976
10.1.11 Project > Archive ... 976
10.1.12 Project > Retrieve ... 977
10.1.13 Project > Analysis ... 978
10.1.14 Project > Exit ... 978
10.1.15 Edit > Cut ... 978
10.1.16 Edit > Copy ... 978
10.1.17 Edit > Paste ... 978
10.1.18 Simulation > Initialize ... 978
10.1.19 Simulation > Start ... 978
10.1.20 Simulation > Pause ... 979
10.1.21 Simulation > Single Step ... 979
10.1.22 Simulation > Exit ... 979
10.1.23 Simulation > Snapshot ... 979
10.1.24 Simulation > Backtracks ... 979
10.1.25 Window > Tile Horizontally ... 979
10.1.26 Window > Tile Vertically ... 979
10.1.27 Window > Unsplit ... 980
10.1.28 Window > Close all ... 980
10.1.29 Automatic model creation > Instantiate templates ... 980
10.1.30 Automatic modelling > Automated import ... 980
10.1.31 Automatic model creation > Bulk engineering import ... 980
10.1.32 Options > Zoom ... 980
10.1.33 Options > SU administration ... 980
10.1.34 Options > Assign coupling signals ... 981
10.1.35 Help > Display help ... 981
10.1.36 Help > About ... 981
10.1.37 Help > SIMIT resources ... 981
10.1.38 Help > License container ... 981
10.1.39 CTE > Component > New component ... 982
10.1.40 CTE > Component > Open ... 982
10.1.41 CTE > Component > Close ... 982
10.1.42 CTE > Component > Save ... 982
10.1.43 CTE > Component > Save as ... 982
10.1.44 CTE > Component > Exit ... 982

10.2 Dialog boxes ... 983
10.2.1 "Available backtracks" dialog box ... 983
10.2.2 "SU Import" dialog box ... 983
10.2.3 "PLCSIM Advanced import" dialog box ... 984
10.2.4 "Virtual controller import" dialog box ... 985
10.2.5 "STL import" dialog box ... 987
10.2.6 "Importing signal properties" dialog box ... 988
10.2.7 "Exporting signal properties" dialog box ... 991
10.2.8 "Select project" dialog box ... 992
10.2.9 "SU administration" dialog box ... 993

10.2.10 "Instantiate templates" dialog box 995
10.2.11 "Bulk engineering import" dialog box 996
10.2.12 "Characteristic" editor 997
10.2.13 "Find & replace" dialog box 997
10.2.14 "Info" dialog box 998
10.2.15 "Print" dialog box 998
10.2.16 "Instantiate template" dialog box 999
10.2.17 "Selection" dialog box 999
10.2.18 "Enumeration" dialog box (CTE) 1001
10.2.19 "Manage components" dialog box (CTE) 1002
10.2.20 "Connection type" dialog box (CTE) 1003

Basics of SIMIT

1

1.1 Operating principle

SIMIT can be used for the following functions:

- Complete plant simulation
  Simulation of signals, devices and plant response
- Input and output simulator of test signals for an automation controller
- Testing and commissioning automation software

Even if only the user interface, for example, is initially used for the signal test, simulation models can subsequently be added whenever required to simulate plant response and carry out dynamic tests.

You essentially work with the following elements to create a simulation:

- **Chart**
  To develop a simulation, you combine the components available in the libraries using the chart editor and enter appropriate passwords.
  You can find additional information in section: Creating and editing charts (Page 252).
- **Visualization**
  Visualization gives you an overview of the signals from your plant. Signal are visualized with controls (input and display objects) and graphical objects.
  You can find additional information in section: Visualizing a simulation (Page 26).
- **Coupling**
  The coupling is the interface to the automation system and is required for signal exchange. You can specify individually the signal scope to be processed by SIMIT.
  You can find more information on couplings in: Couplings (Page 55).

1.2 DEMO mode

In DEMO mode you can get an idea of the handling and performance of SIMIT.

Starting SIMIT in DEMO mode

Start SIMIT DEMO with a double click on the screen link SIMIT SP Demo. A dongle is not necessary.

Only the following SIMIT function modules are available in DEMO mode:

- Macro component editor
  You can find more information in section: Macro component editor (Page 271).
- Chart editor
  You can find more information in section: Creating and editing charts (Page 252).

- Trend and messaging editor
  You can find more information in section: Trend and messaging editor (Page 309).
- Automatic model creation
  You can find more information in section: Automatic model creation (Page 279).

SIMIT has only limited functionality in the following areas in DEMO mode:

- **Virtual controller**
  You can run SIMIT in DEMO mode and the virtual controller licensed or vice versa. The DEMO mode restrictions however apply both to SIMIT and to the virtual controller.
- **Saving and archiving**
  Projects, templates and macro components can be saved in DEMO mode. Projects, templates and macro-components created in DEMO mode can, however, only be used on the computer on which they were created.

  Note

  Projects, templates and macro-components created in DEMO mode are not compatible with the full version of SIMIT.

  Projects created in DEMO mode cannot be upgraded.

  You cannot archive projects in DEMO mode.

- **Opening and retrieving**
  You can open a project in DEMO mode under the following conditions:
  - Project was created on this computer in DEMO mode.
  - Project version corresponds to the installed software version.

  Projects created with a full version cannot be opened.
  You can retrieve projects that have been archived in a full version. However, if the retrieved project is changed in DEMO mode, it cannot then be used in the full version.
- **Address area**
  In DEMO mode, a maximum of 30 coupling signals can be connected in the simulation as long as a maximum of 30 bytes each is covered in the input area and the output area.
- **Runtime**
  You can use SIMIT in DEMO mode for as long as you want, but the runtime of a simulation is limited to 45 minutes. The simulation ends automatically at the end of those 45 minutes. You can restart the simulation once it has ended.
- **Number of couplings**
  The "External couplings" programming interface is not supported.
- **Project folder**
  In DEMO mode, you can only save projects in a specified memory location in the SIMIT work area.
- **Libraries for macro components and templates**
  In DEMO mode, you can only store macro components and templates within the SIMIT work area. You cannot open other library folders.
- **Snapshots**
  Snapshots cannot be taken in DEMO mode.

- **Simulation process**
  In DEMO mode, the simulation can only run in real time. It cannot run faster or slower.
- **Number of virtual controllers**
  You can run a maximum of 32 virtual controllers per SIMIT system.
- **Size variant**
  You can only use the size variant S in DEMO mode.

1.3 Starting SIMIT

Start SIMIT either using the Windows Start menu "**All Programs > SIMIT > SIMIT Simulation Platform**" or by using the link on the screen.

When starting SIMIT, select the size variant (XS, S, M, L, XL). You can also change the size variant later via Portal view > Start > Settings > Size variant.

The portal view or project view opens after startup. You set which view is opened via Portal view > Start > Settings.

The clear layout of the portal view helps you to quickly become familiar with SIMIT. The most important basic functions can be selected directly:

- Manage projects
- Create couplings
- Create simulation models
- Automatic data import from different file formats

Additional functions are:

- Perform consistency checks
- Accessing the "Find & replace" function
- Create a new trend
- Process archives

Click "Project view" to change to the project view. Full functionality of SIMIT is only available in the project view.

Getting started

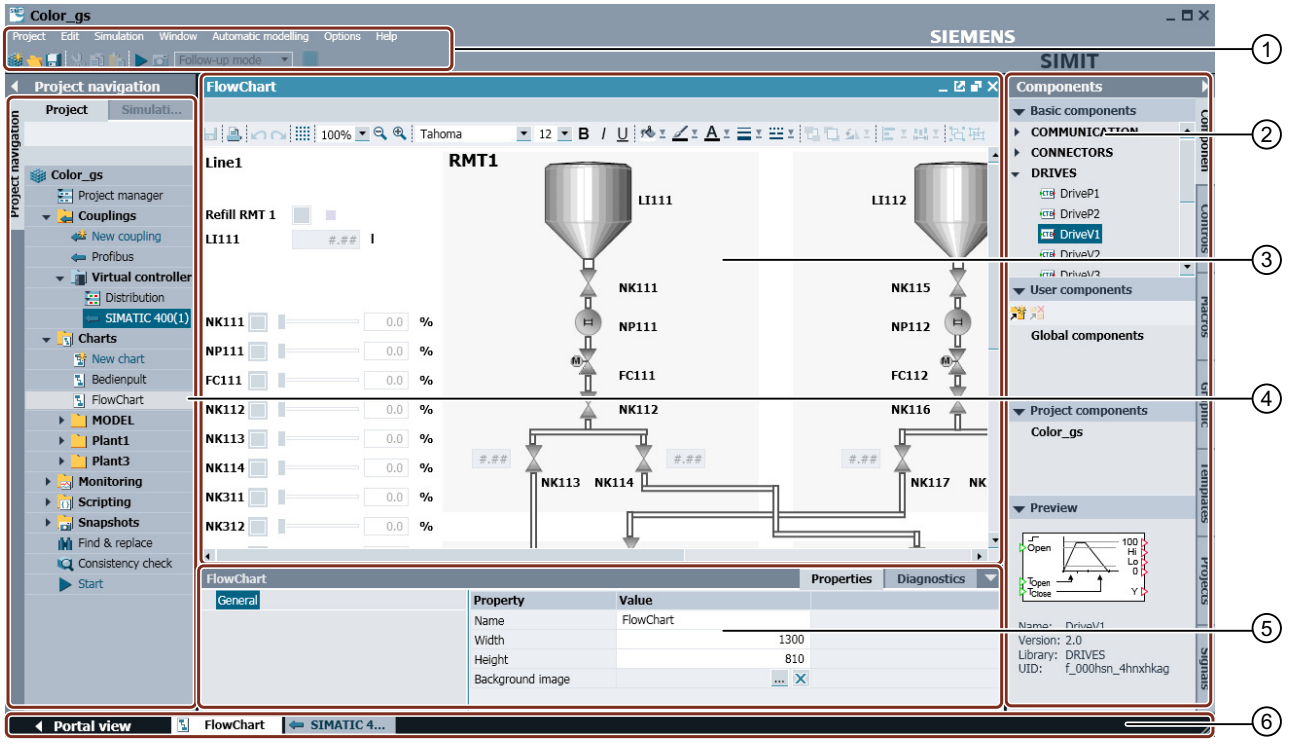
After opening, creating or retrieving a project, the "Getting started" section opens. This is where you can add the basic objects that are required for the function of a SIMIT project.

- Add coupling
- Add chart
- Generate charts automatically

You can perform these steps here or go to the **project view** to continue editing the project and start the simulation.

1.4 The SIMIT graphical user interface

The user interface of SIMIT is divided into the following areas:



①	Menu bar and toolbar	Access to SIMIT functions. Further functions are available in the shortcut menus in the project navigation.
②	Task cards	Objects such as library components, controls and graphic objects that can be used in the editor currently open are listed here. These objects are sorted in task cards.
③	Work area	This is where the editors are opened for editing.
④	Project navigation	The current project is displayed here in a tree view.
⑤	Property view	The properties of the selected object are displayed here.
⑥	Editor bar and status bar	Here, you can switch between open editors and the portal view. If necessary, information on the current status of SIMIT is displayed here.

All editors are opened in the work area. Task cards are provided specifically for each editor. The work area can be divided in order to open two editors side-by-side or underneath one another in the work area.

The window buttons of an editor:



The following functions can be performed using the window buttons of an editor:

- Minimize, i.e. reduce the entry in the editor bar
- Reduced to occupy only part of the work area

- Maximized to occupy the entire work area
- Close
- Detach an editor with task cards and property view from the work area as a separate window
- Insert an editor window that was detached from the work area back again into the work area
- Anchor a detached editor window in the foreground

Project navigation

Projects are managed in the project navigation. A SIMIT project is divided into the following tree nodes, whereby a project must not contain all of the objects listed here:

Color_gs	Current project name
Project manager	Tree node for the project manager. Elements from other projects can be copied into the currently opened project in the project manager. You can find additional information on this in section: View and functions of the Project Manager (Page 247).
Couplings	"Couplings" folder: Folder for storing the couplings of the current project. You can find additional information on this in section: Couplings (Page 55).
	Symbol for the couplings. Couplings create the connection between SIMIT and a controller or another application.
New coupling	Tree node to create a new coupling.
Charts	"Charts" folder: Folder for the charts.
	Symbol for the charts. Charts contain a simulation model created using library components and controls.
New chart	Tree node to create a new chart. You can find additional information on this in section: Structure of a simulation (Page 24).
Monitoring	"Monitoring" folder: This folder contains the following functions and is also their storage location: • New trend • Messages • Archive You can find additional information on this in section: Functions of the Trend and Messaging Editor (Page 309).
Scripting	"Scripting" folder: This folder contains the "New script" function and is the storage location for existing scripts. You can find additional information on this in section: Creating a script (Page 334).

	"Lists" folder: This folder contains the "New list" function and is the storage location for existing material lists. This folder is only relevant if the "CONTEC" library has been installed. You can find additional information on this in section: Modeling of the objects (Page 871).
	"Snapshots" folder: Folder for storing snapshots of the simulation When simulation is running, the folder also contains the "New snapshot" function. You can find additional information on this in section: Actions during ongoing simulation (Page 34).
	Tree node for "Find & replace" function Searches can be made for signals, components, connectors, and graphic texts. You can find additional information on this in section: Find & replace (Page 323).
	Tree node for performing the consistency check. The consistency check verifies the project for formal errors. You can find additional information on this in section: Consistency check (Page 328).
	Tree node for starting simulation.

Each of these functions is run with a double-click.

Functions for the individual project elements are also available in the shortcut menu for the tree entry.

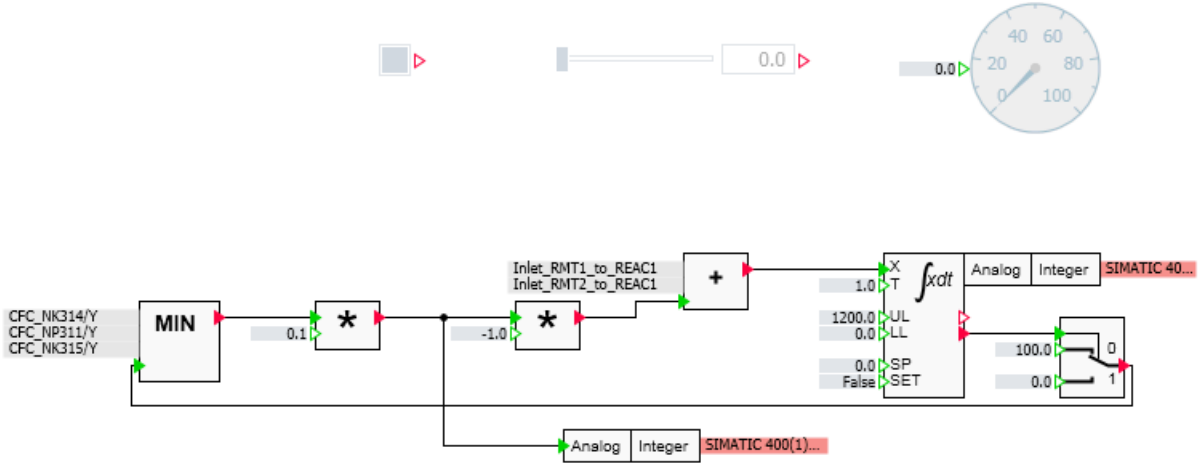
1.5 Creating a simulation

1.5.1 Structure of a simulation

A simulation is created in charts with components, controls and connections and is linked to the coupling with connectors.

You can find a brief description of these elements below. For more information, please see the links.

- **Chart**
  The chart contains the simulation model. Components, controls and connections are graphically displayed. Charts are created and edited in the chart editor. A simulation model can consist of multiple charts.
  You can find additional information in section: Chart editor (Page 252).
  The figure below shows an example of a chart with components, connections and controls:



Charts consist of the following objects:

- **Components**
  You will find the components for logical and arithmetic functions and for drives, sensors, connections and communication on the "Components" task card under "Basic components". To add components to a chart, select them and drag-and-drop them to the chart.
  You can find additional information in section: "Components" task card (Page 261)
- **Controls**
  You will find the controls for entering and displaying values on the "Controls" task card. The "Display" area contains objects for the dynamic display of values from a current simulation. The "Input" area contains objects for specifying values in a current simulation.
  To add controls to a chart, select them and drag-and-drop them to the chart.
  You can find additional information in section: Controls (Page 584)
- **Connections**
  Connections are defined with signals and shown in the chart either with direct connections or as connectors.
  Signals connect components and controls to the coupling and the coupling to the automation system. The controller signals are processed in the coupling editor.
  - **Connectors**
    Signals are shown on a chart by connectors: Output signals are shown as green output connectors *(Output)*, input signals are shown as red input connectors *(Input)*. Drag-and-drop connectors from the coupling to charts. Split the work area with the menu command "Window > Split horizontally" and open the coupling and the chart. Drag the required signal from the coupling to the chart by selecting it in the coupling window at the left margin and holding down <Shift>. Connect the connector I/O to the I/O of a component. Coupling signals can also be dragged to a chart from the "Signals" task card. Filter the signals accordingly, hold down <Shift> and drag the required signals to the chart.

To create a new chart, drag the components and controls from the relevant task cards to the chart, connect their I/O and enter the parameter values. You can find more information in section: Creating and editing charts (Page 252).

- **Coupling**
  Couplings are used by SIMIT to communicate with the automation system. Coupling signals can be imported in various formats and can be edited in the coupling editor.
  A coupling is created in the project navigation by double-clicking on the tree entry "New coupling". Select the required coupling type and double-click again on the new coupling to open the coupling editor. Here, you can import and edit signals.
  You can find more information in section: Couplings (Page 55).

Note

The coupling configuration must first be saved before a signal can be extracted from it.

1.5.2 Visualizing a simulation

The signal values and signal states of a simulation are shown with graphics and controls.

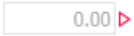
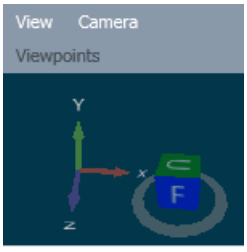
Controls are ready-to-use objects for entering or displaying signals. They just need to be connected to the required signal.

Graphics are created individually. Graphics can be static or animated.

Controls

Controls are categorized by signal input or display and by data type. The following controls are available:

Name and symbol	Data type	Use	Link to additional information
Binary display 	Binary	Signal display	Binary display (Page 584)
Analog display 	Analog, integer	Signal display	Analog display (Page 585)
Digital display 	Analog, integer	Signal display	Digital display (Page 586)
Bar graph display 	Analog, integer	Signal display	Bar graph display (Page 588)
Button 	Binary	Signal input	Pushbutton (Page 590)
Button with image 	Binary	Signal input	Pushbutton with image (Page 591)
Switch 	Binary	Signal input	Switch (Page 592)
Switch with image 	Binary	Signal input	Switch with image (Page 593)
Step switch 	Integer	Signal input	Step switch (Page 594)
Step switch with image 	Integer	Signal input	Step switch with image (Page 595)

Name and symbol	Data type	Use	Link to additional information
Digital input 	Analog, integer	Signal input	Digital input (Page 596)
Slider 	Analog	Signal input	Slider (Page 598)
3D viewer 	–	Integrating three-dimensional graphics into a chart	3D Viewer control (Page 602)
Signal isolator	–	Fixing signals, must be connected to other controls	Signal splitter (Page 600)
Action	–	Opening charts and trends.	Action (Page 602)

Proceed as follows to integrate a control into the current chart:

1. Select the "Controls" task card.
2. Click on the required control.
   Information on the control is displayed in the "Preview" area.
3. Drag-and-drop the control to the chart.
4. Interconnect the control.
   You have the following options:
   - Connect the control directly to a component.
   - Connect the control over a connector.
   - Connect the control over a coupling signal from the "Signals" task card.

Graphics

You use graphics to display signals with an individual view. Static graphics specify a particular display; animated graphics change within the simulation in size, color and position in line with their current state.

The following static graphics are available:

Graphic element	Brief description	Settings
Text	Entry of text at any point in the chart.	Font Font size Font color Background color Border color
Line	Drawing a straight line. Hold down <Shift> to create a horizontal or vertical line.	Line color Line width
Rectangle	Drawing a rectangle. Hold down <Shift> to create a square.	Border color Border width Fill color
Ellipsis	Drawing an ellipsis. Hold down <Shift> to draw a circle.	Border color Border width Fill color
Polyline	Drawing a polyline. Double-click or press <Space> to finish drawing.	Line color Line width
Ellipse arc	Drawing an ellipse arc. Hold down <Shift> to create an arc.	Line color Line width
Bezier curve	Drawing a Bézier curve	Line color Line width

Proceed as follows to integrate a graphic into the current chart:

1. Select the "Graphics" task card.
2. Click on the required graphic.
   Information on the graphic is displayed in the "Preview" area.
3. Drag-and-drop the graphic to the chart.
4. Interconnect the graphic.
   You have the following options:
   - Connect the graphic directly to a component.
   - Connect the graphic over a connector.
   - Connect the graphic over a coupling signal from the "Signals" task card.

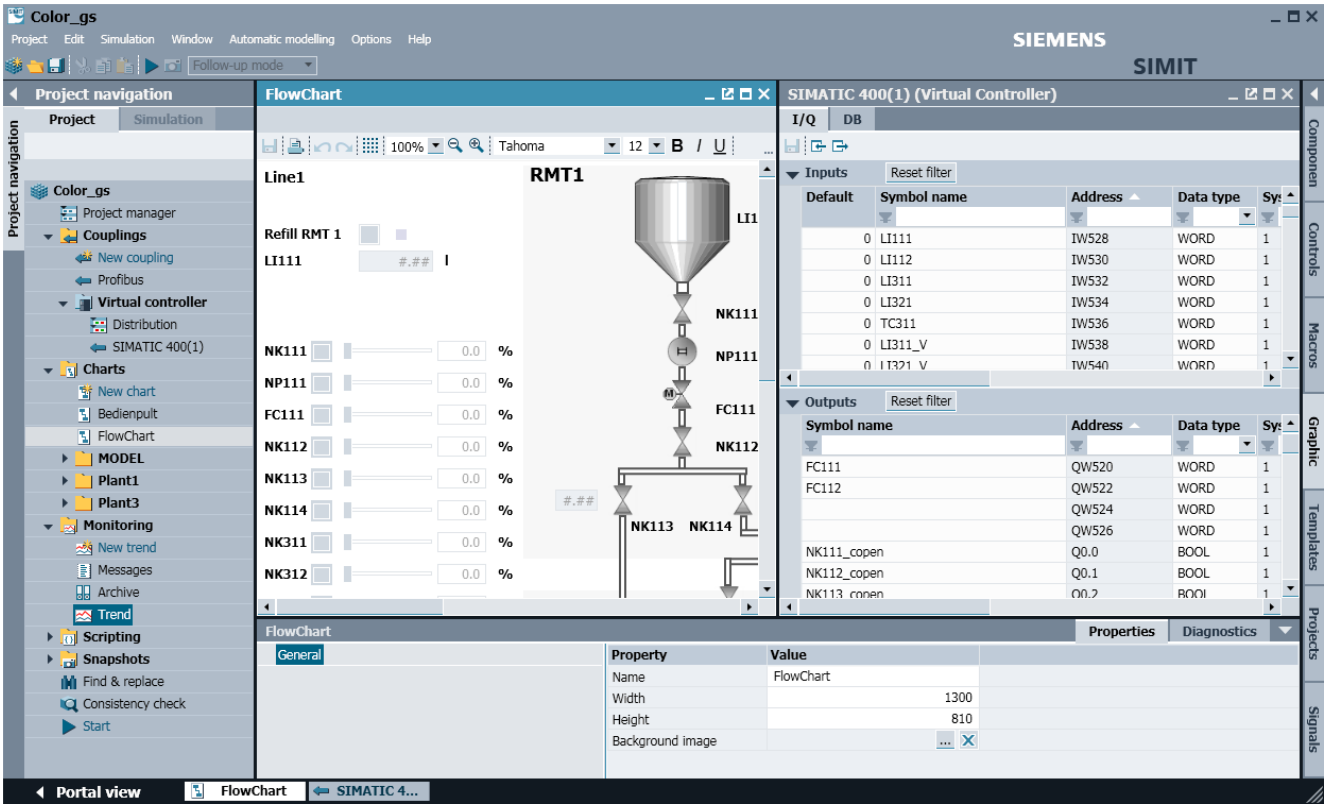
You can find information on using animated graphics in the section: Visualizing graphics (Page 254)

1.5.3 Visualizing coupling signals

When a simulation is running, a control with a signal splitter is provided for each signal in the coupling. Proceed as follows to position this control in a chart:

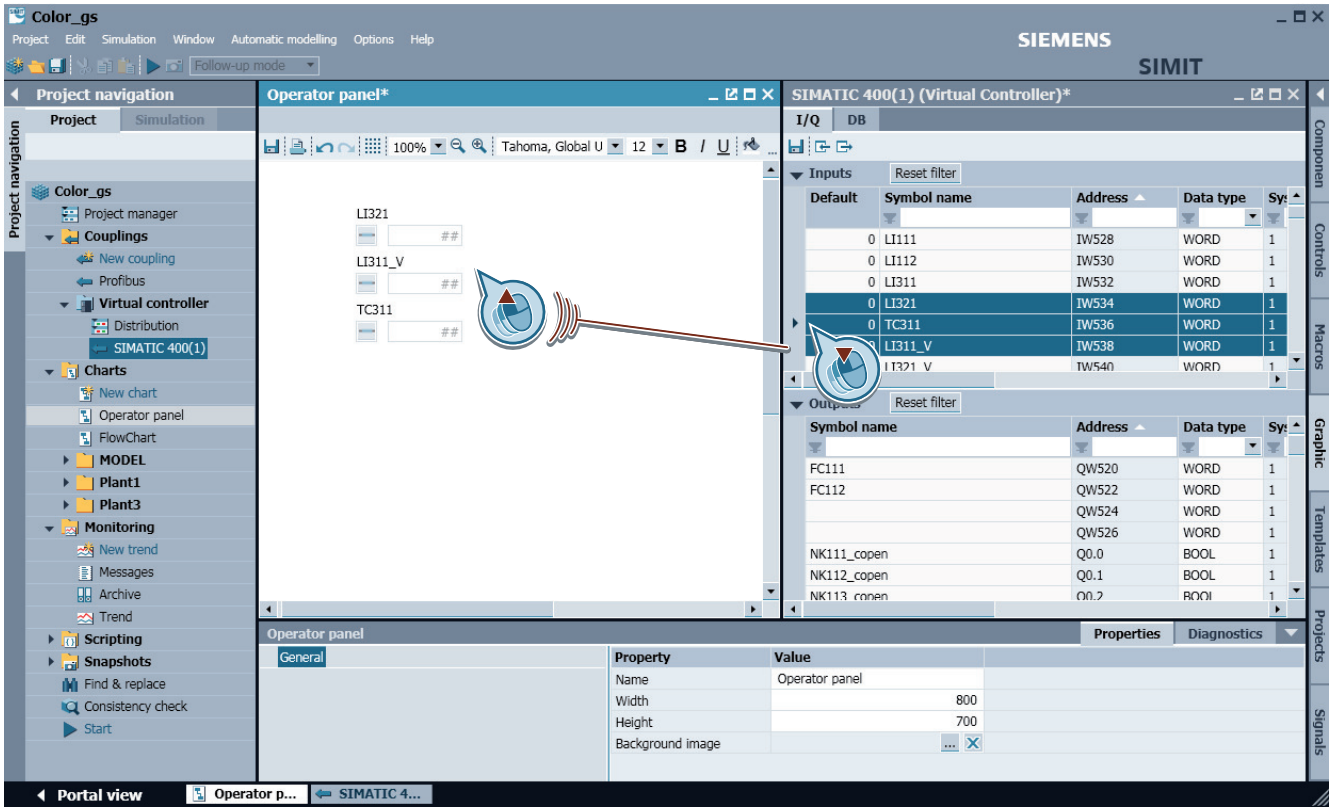
1. Split the work area with the menu command "Window > Split horizontally".
2. Open the coupling and the chart in the work area.
3. Create a new chart.

4. Open the chart.
5. Open the coupling.
   You see the chart and coupling in the work area as shown in the figure below:



6. Set appropriate filter settings in the coupling window and select the required signal with a click of the mouse.
   To select multiple signals, press <Strg> (individual selection) or <Shift> (area selection).

7. Select the "▶" symbol in the first column of the coupling window.
8. Drag-and-drop the selected signals to the chart as shown in the figure below:



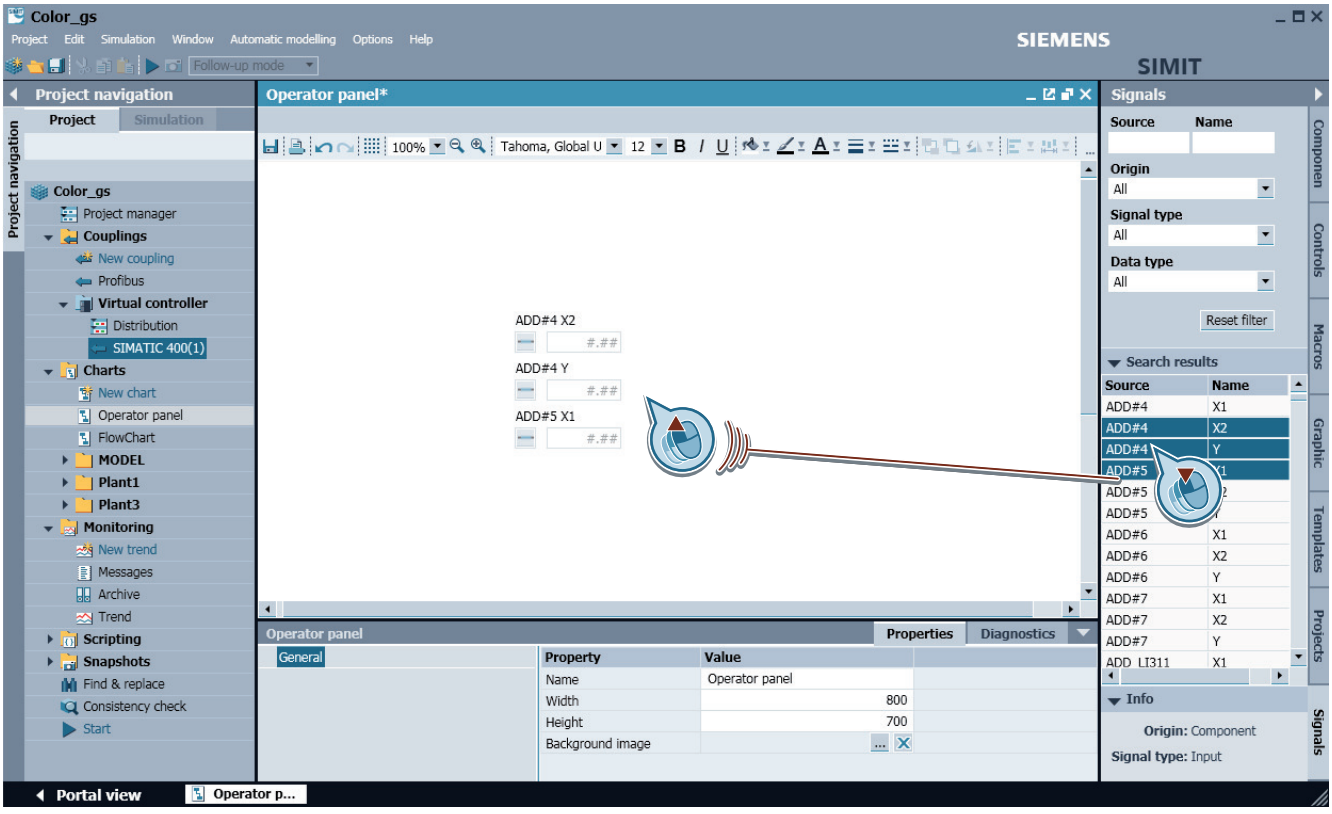
A switch is generated for signals with the data type "Binary", and a digital input for all other signals, as a control in the coupling.

Dragging coupling signals from the "Signals" task card to the chart

You can alternatively also drag coupling signals from the "Signals" task card to charts. Follow these steps:

1. Open the chart in the work area.
2. Open the "Signals" task card.
   All signals saved in the project are available on this task card.

3. To select signals, set corresponding filters for the signal list. You can find additional information in section: "Signals" task card (Page 270).
4. Select the signals from the list and drag-and-drop them to the chart as shown in the figure below:



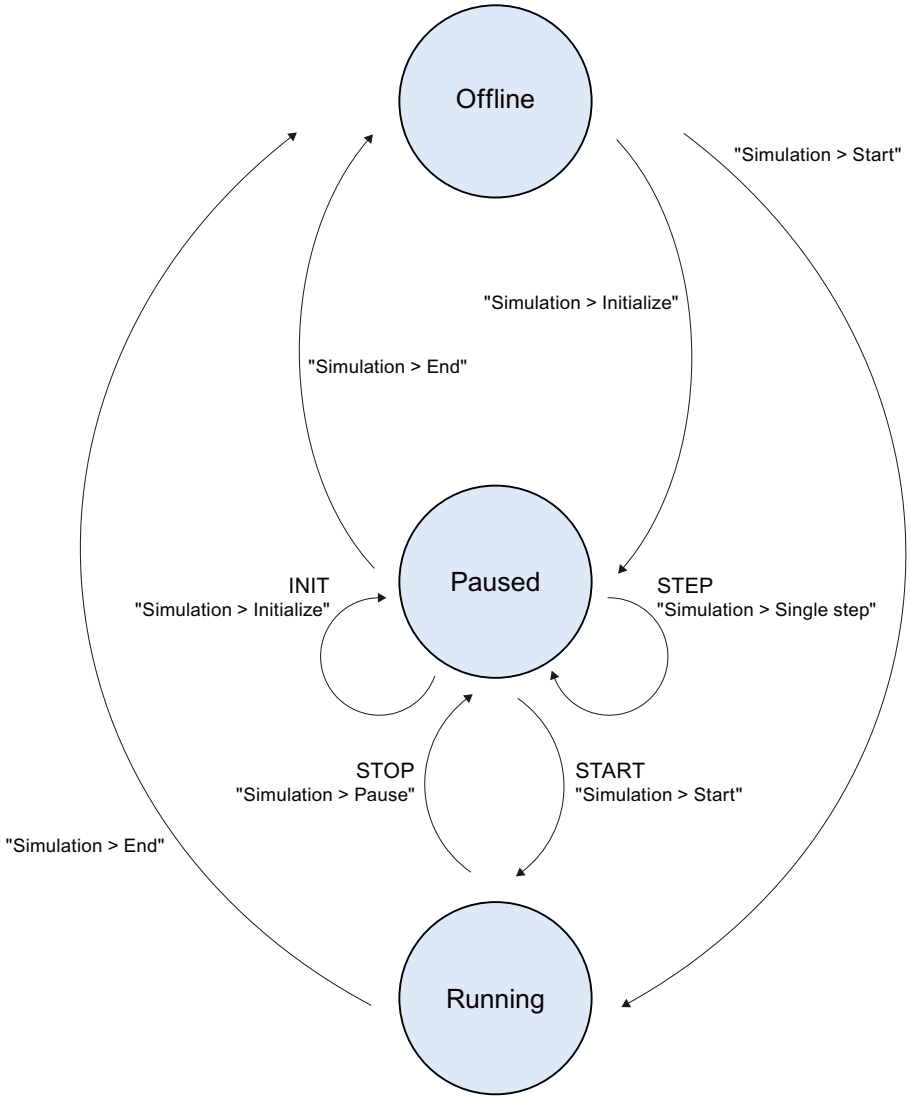
1.6 Simulation

1.6.1 The states of a simulation

A simulation is always in one of the following states:

- Offline
- Paused
- Running

These states are changed via the "Simulation" menu or the toolbar. The following graphic shows the states and the related commands that are used to change from one state to another:



Initializing simulation

Simulation is initialized with the menu command "Simulation > Initialize".

You can run simulation initialization before starting a simulation. The state changes to "Paused". The initialization runs the consistency check; messages are displayed in the event of an error. You can find additional information on this in section: Consistency check (Page 328).

The "Single step" function can be executed by an initialized simulation.

Simulation starts

Simulation is started with the following commands.

- Button "▶" in the toolbar
- With the menu command "Simulation > Initialize".
- With the project navigation by double-clicking on ▶ Start

Once simulation has been started, SIMIT changes the color scheme of the interface from *blue* to *orange*; the state changes to "Running".

You can find additional information on running simulation in: Actions during ongoing simulation (Page 34).

Pausing the simulation

Simulation is paused with the following:

- Menu command "Simulation > Pause"
- Button  in the toolbar. The "Start" button changes with the "Pause" button

A simulation can be paused at any point. The simulation is paused in its current state until it is started again or ended.

When using the SIMIT Unit, pausing the simulation for more than 30 seconds terminates the connection between the SIMIT Unit and SIMIT.

Executing a single step

A single step is only available if the simulation is in the "Paused" state.

A single step can be executed with the following commands:

- With the menu command "Simulation > Single step".
- With the <F12> function key

An single step executes the smallest time slice of the project and then pauses until another step is triggered or the "Start" or "End" command is executed.

Exiting simulation

Simulation is exited with the following:

- Button "" in the toolbar
- Menu command "Simulation > Exit"

The state changes to "Offline" and the color scheme of the interface changes back to *blue*.

1.6.2 Simulation operator control and monitoring

1.6.2.1 Actions during ongoing simulation

You can execute the following actions while simulation is running:

- Opening and closing charts and couplings
- Monitoring value changes at the inputs and outputs of selected components in the property view
  Select the "" icon in the toolbar to display value changes of outputs and interconnected inputs in the chart. The value changes are also displayed in the In the properties of the component.
- Generating and displaying a trend
  You can find additional information in section: Trends (Page 314)

- Creating a script
  You can find additional information in section: Handling of scripts (Page 334)
- Saving a snapshot
  A snapshot is created with one of the following commands:
  - With the menu command "Simulation > Snapshot"
  - With the "" symbol in the toolbar
  - With a double-click on the entry "" in the project navigation

  The snapshot saves the current state of the simulation. Each snapshot is saved in the "Snapshots" () folder in the project navigation and is named with the date and time of saving. You can rename a snapshot, move it to a subfolder or delete it.
  If you load a snapshot, the simulation is set to the state saved in the snapshot and the simulation is continued from that state. This process automatically starts the simulation.

The color scheme of the user interface changes to *orange* in the "Paused" and "Running" states.

1.6.2.2 Display of the simulation load

Simulations are calculated cyclically. The cycle time is set with a time slice. Each component of a chart and each coupling is assigned to one of multiple available time slices. The solution procedures are calculated in the time slice in which the associated components are involved. The absolute cycle time for a time slice is set in the project manager.

The cycle time specifies the time base in which the calculations are to be run, e.g. every 100 ms. If not all the calculations of the time slice can be executed within the cycle time, however, this time frame cannot be met.

SIMIT determines the simulation load to assess the degree to which the computer is busy. It indicates the percentage ratio of actual computation time for the configured cycle time. This allows you to establish whether the cycle time is sufficient for all calculations. Only the worst value overall is displayed.

Example:

	Time slice 1	Time slice 2
Configured cycle time	100 ms	200 ms
Calculation time	60 ms	80 ms
Simulation load	60%	40%
Total simulation load (load value)	60%	

Meaning of the percentage load value:

- **Load value significantly less than 50%**:
  The simulation can be calculated in the set cycle times.
  The data is exchanged as scheduled between the simulation model and the couplings.
- **Load value reached or exceeds 50%:**
  The simulation can be calculated in the set cycle times.
  The data is no longer exchanged with a time delay between the simulation model and the couplings. Additional cycle offsets are created.
  The simulation values are not transferred to the couplings or read from the couplings until a later computing step.
  If a simulation project contains no couplings, the 50% limit is irrelevant.
- **Load value reached or exceeds 100%:**
  The simulation is in overload.
  The simulation model can no longer be calculated within the planned cycle time.
  Computational steps are omitted.

The simulation load is shown while simulation is running via an symbol in the low right pane. The current simulation load is shown by the green segment (full circle equals 100%):



Load fluctuations

The load value is not always constant. A fluctuating load can be caused by:

- The simulation model itself can create more load in each calculation step due to the configuration
- A time slice is interrupted by a time slice with a higher priority.
- Other applications are running in the background in addition to SIMIT
- Internal processes in both the operating system and in the .NET environment used by SIMIT may delay the simulation calculation

In order to keep the graphical representation of the interface calm, the load values are smoothed over several increments.

Note

If the load value always jumps back and forth between two values, the problem may be that the processor of your computer keeps changing the clock frequency for energy saving reasons. You can avoid this by altering the system settings of your computer, for example. Select "Control panel > Power Options > Power plan > Advanced power settings > High performance".

Note

The Windows Task Manager is not suitable for evaluating the simulation load. On the one hand, the distribution of the simulation model on multiple processors or cores has to be taken into consideration; on the other hand, the task manager only indicates the computing power but not whether the configured cycle time cannot be met due to communication delays.

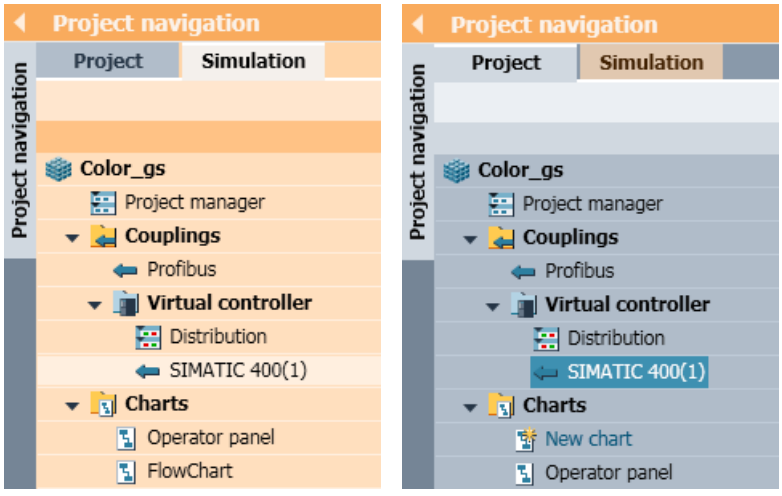
Note

If the simulation load is too high, you can access information on the cause with the "SimulationLoad" component.

You can find additional information on the component in: SimulationLoad (Page 495)

1.6.2.3 Changes in a simulation project while a simulation is running

You can make changes to the simulation project while the simulation is running. To make changes while the simulation is running, toggle between "Simulation" and "Project" in the project navigation.



You can open charts in both project views, but for different purposes:

- If you are in the blue project view, charts are opened for editing. The changes that you make take effect the next time the simulation is started or when you select the "Activate changes" menu command.
- If you are in the orange simulation view, charts are opened for operation. Their content corresponds to the simulation that is running and cannot be changed.

The same chart can be opened simultaneously in both modes. When the simulation ends, the chart opened from the simulation tree closes automatically.

While the simulation is running, the following restrictions apply to using SIMIT:

- No new couplings can be created and existing ones cannot be deleted.
- Only implicit interconnections can be edited in the couplings.
- All functions of "Automatic modeling" are disabled.

Opening charts offline

You can navigate directly from a chart in online mode to the corresponding chart in offline mode.

1. Click on an empty point on the chart in online mode.
2. Select "Open offline" in the shortcut menu. The system navigates to the chart in the blue project window in the project navigation.
3. Open the chart.

You can make changes in this chart without having to stop the simulation.

Note

The "Open offline" function is possible when:

- The chart has the same name in online and offline mode.
- The chart is in the same folder hierarchy in both modes.

Preparing and activating model changes

Model changes can also be activated during the simulation.

Follow these steps:

1. Switch to the project view while simulation is running.
2. Make the desired changes to the simulation model.
3. Click "Prepare changes". The preparation may take several minutes.
   Once the "Prepare changes" function is completed, the menu command changes to "Activate changes".
4. To apply the changes to the simulation in progress, click "Activate changes".

Note

While the changes are activated, the simulation model does not process any values for some time. During this time, I/O devices may not respond. The length of time depends on the size of the simulation model.

If the current simulation state is transferred to the new configuration, the values can be identified and assigned based on the signal name. Values for which there is no equivalent are assigned initial values.

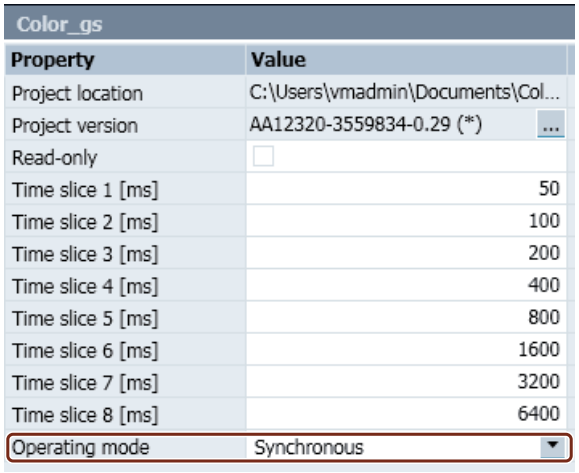
Activation does not include the values of parameters and defaults that are modifiable online; the values from the simulation in progress are retained. Restart the simulation to enable these value changes.

1.6.3 Settings for simulation

1.6.3.1 Operating modes

The simulation model and the couplings can be processed according to the three different operating modes *asynchronous*, *synchronous* and *isochronous*.

Which mode is appropriate depends on the project. It is set in the property view in the project manager:



For additional information about modes, refer to the sections:

- Asynchronous operating mode (Page 39)
- Synchronous operating mode (Page 41)
- Isochronous mode (Page 42)

Asynchronous operating mode

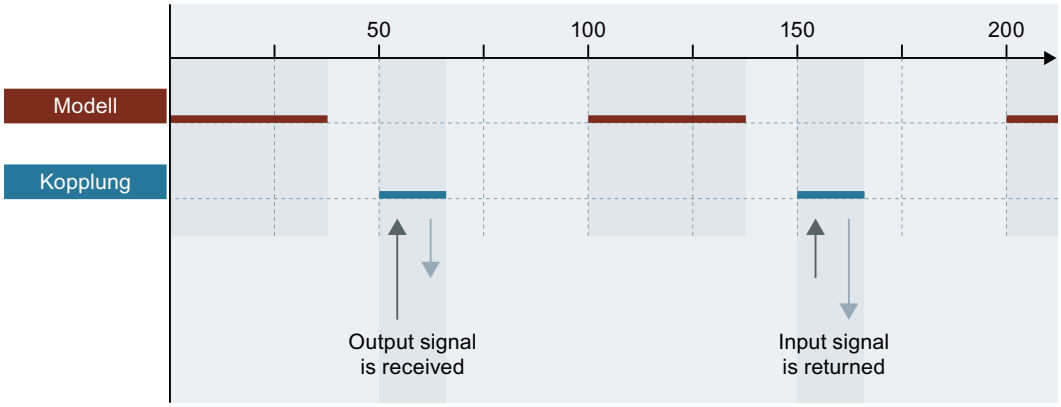
Model calculation of the individual time slices and coupling processing is time-controlled.

If the simulation model of a time slice does not finish within the allocated time, one or more processing cycles are omitted and the time slices with other cycle times are not calculated.

The calculation of the simulation model only continues when all time slices can calculate once again.

If SIMIT does not calculate a coupling within the allocated time, one or more processing cycles are also omitted here but that does not prevent calculation of the couplings in time slices with other cycle times and the calculation of the simulation model. This means if couplings are blocked by a communication problem, this does not affect the overall processing.

In order to achieve the fastest possible response times, couplings are scheduled with a time offset compared to the model calculation:



If the model calculation or the data communication of couplings takes longer than half of the cycle time, this time offset does not result in faster response time.

Signal exchange via couplings in asynchronous mode

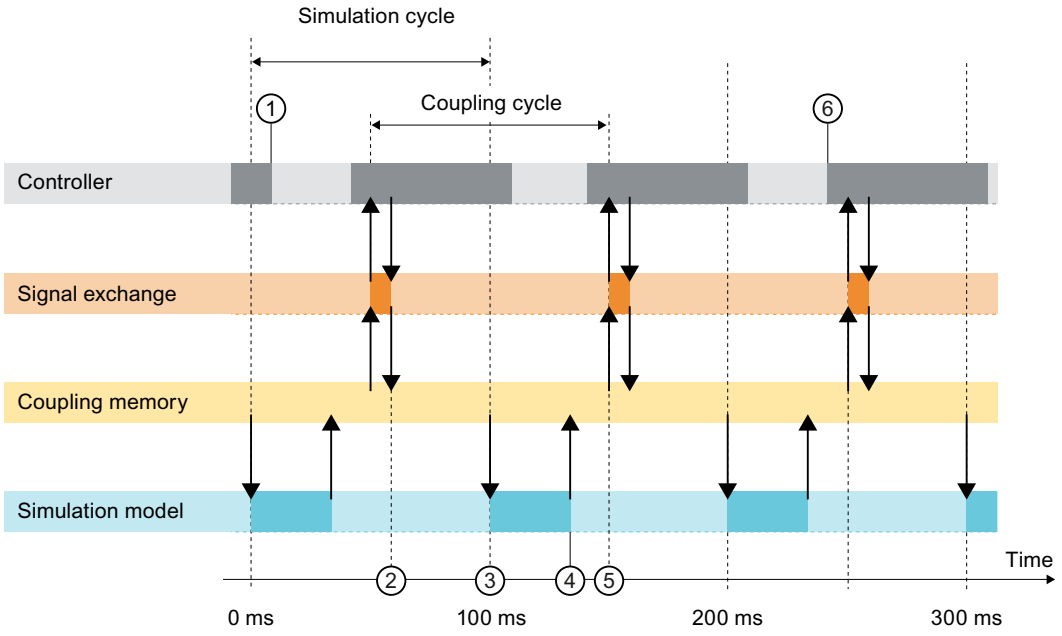
Both the calculation of the model and the exchange of signals in the coupling are triggered cyclically by the control system. The cycle time for the model calculation is specified in the components or in the project properties. The cycle time for the exchange of signals is specified in the coupling properties (coupling configurator) regardless of this. Delays in the exchange of signals in a coupling therefore have no effect on the model calculation but only on the other couplings that are processed in the same cycle time.

To keep delays in the exchange of signals as small as possible, the time at which the coupling is triggered is generally shifted by half the set simulation cycle relative to the time at which the simulation model is triggered.

Note

The same value should be set for both the cycle time of the coupling (coupling cycle) and the cycle time for the calculation of the simulation model (simulation cycle).

In the example below, both the automation system and the calculation of the model have a cycle time of 100 ms. As the automation system and the simulation are not synchronized, their timing is usually shifted with respect to each other. The simulation cycle takes no longer than half the model calculation cycle:



① At this time, the automation system has finished its calculation cycle. Additional signals are available at the outputs of the automation system.

② The output signals of the automation system are applied to the SIMIT coupling memory. Here they are available for the calculation of the model.

③ The calculation of the model is triggered again.

④ End of the model calculation cycle. The calculated input signals for the automation system are now available in the coupling memory.

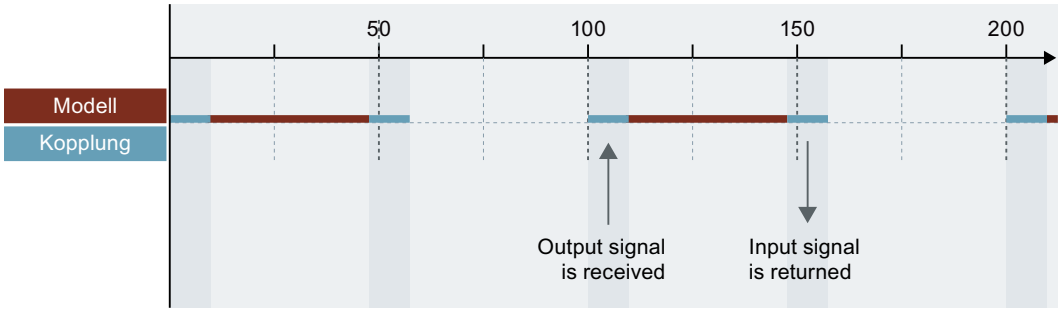
⑤ With the next clock pulse of the coupling, these input signals are transferred to the controller.

⑥ If the controller only evaluates its inputs at the start of a cycle, the transferred input signals are only used in the control program now. This means the reaction of the simulation does not take place until two control cycles later.

Synchronous operating mode

With synchronous mode, modules and couplings are calculated in a precisely specified sequence. The next action does not start until the previous action has finished.

The coupling is divided into output signal recording and input signal writing:



This results in better response times with short cycle times. However, each module and each coupling can block the entire simulation execution.

Signal exchange via couplings in synchronous mode

In synchronous mode, model calculation and signal exchange is performed in a specified sequence. The coupling detects the output signal, and writes back the input signal only after a model calculation. In contrast to asynchronous mode, no control cycles can be dropped.

Isochronous mode

With the isochronous operating mode, SIMIT ensures that all components involved in the simulation have the same synchronized simulation progress. This operating mode is therefore suitable for applications with real-time requirement. For applications without real-time requirement, asynchronous or synchronous operating mode is sufficient.

Requirement:

- Isochronous mode is available for the couplings PLCSIM Advanced and MCD.
- In isochronous mode, a time slice must correspond to the clock time.

In isochronous mode, only the following control commands are available from the "Simulation" menu:

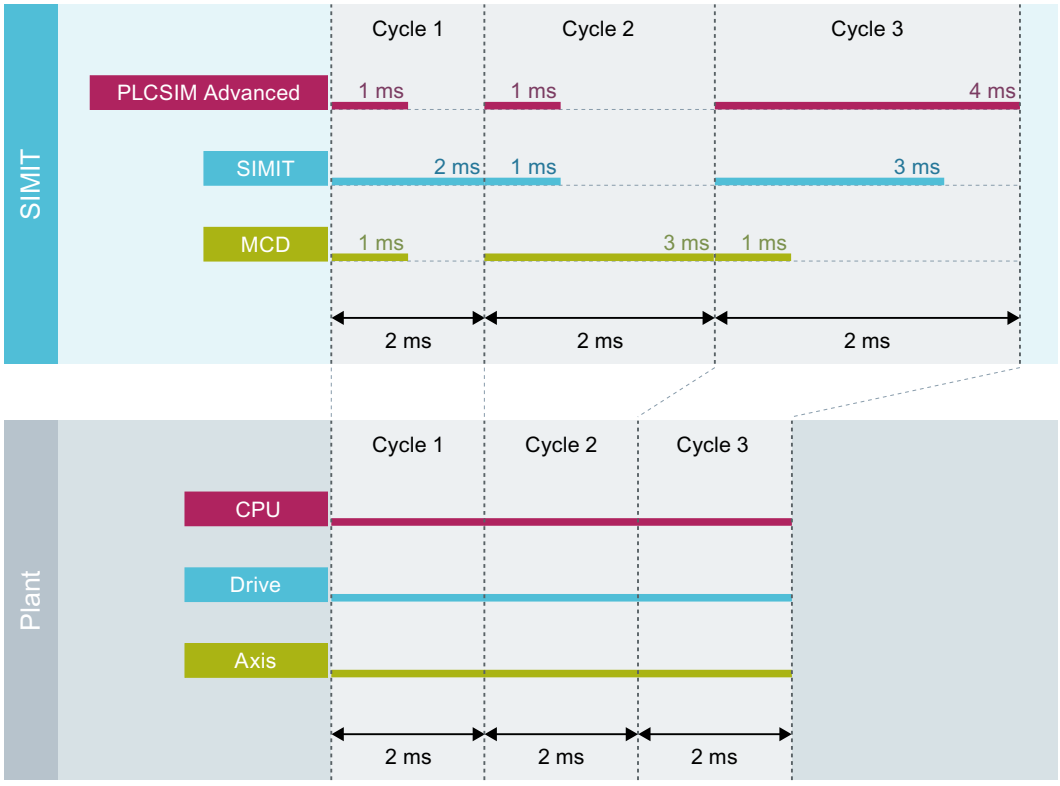
- Start
- Close

Example

In the example shown here, the cycle time is 2 ms and a time slice of the project must be set to 2 ms.

In the top part of the figure you can see how long PLCSIM Advanced, SIMIT and MCD need for their calculations. Only when all three calculations are complete and the results have been exchanged do the calculations for the next cycle start. Even if the calculations in the simulation take longer than the cycle time of 2 ms, the calculation time in the real plant always corresponds to the set cycle time.

In the bottom part of the figure you can see the timing of the CPU, drive and axis in the real plant.



1.6.3.2 Time slices

Definition

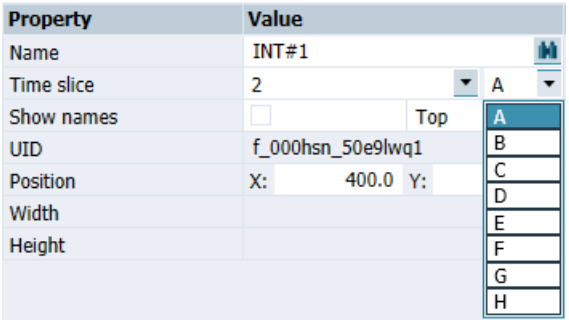
Simulations are calculated cyclically. The cycle time specifies the time frame in which the calculations are to be performed and data is to be exchanged. The cycle time is set by means of one of eight available time slices. The absolute cycle time for a time slice is set in the project manager and is valid for the entire project. The minimum cycle time that can be set is 1 ms.

Time slices of charts

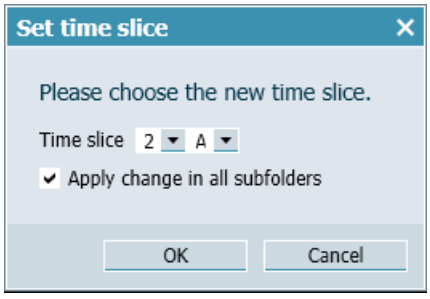
Each component of a chart is assigned a time slice. The solution procedures are calculated in the time slice in which the associated components are involved. The cycle time specifies the time frame in which the calculations are to be performed, for example, every 100 ms. If not all the calculations of the time slice can be executed within the cycle time, however, this time frame cannot be met.

Components of a chart have eight available sub-time slices, A to H, for each time slice. All sub-time slices run in the cycle time of the associated main time slice. The sub-time slices offer a better distribution of the simulation to better utilize multi-core processors.

You set the time slice and the sub-time slice of a component of a chart in the properties of the component under "General".



You set the time slice and the sub-time slice of a chart or chart folder using the shortcut menu in the "Set time slice" dialog box.



Time slices of couplings

Each coupling is assigned a time slice. The time slice determines the cycle at which the coupling exchanges data. Couplings always run in the main time slice.

You set the time slice of a coupling in the properties of the coupling.

You set the time slice of a coupling folder using the shortcut menu in the "Set time slice" dialog box.

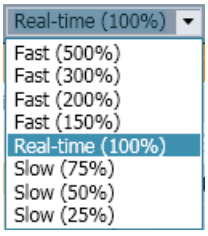
Note

Note that a maximum of 7 time slices in total can be used for all couplings.

1.6.3.3 Speeding up and slowing simulation

To accelerate or slow the progress of simulation, you can change the ratio of simulation time to actual time.

This value is set via a drop-down list in the toolbar:



Maximum acceleration is limited in such a way that the shortest cycle time of the project is at least 1 ms taking into account the acceleration. After the end of the time shift, the time is synchronized with the computer time again.

This means acceleration may not be possible depending on the cycle time. Only suitable values are offered for selection.

Note

Time response

Timer OBs in the virtual controller and timer functions that are dependent on timer OBs in the virtual controller may result in an irregular time response.

1.6.3.4 Adjustable timeout times

Various operations are monitored by SIMIT and terminated if they exceed a pre-defined time in order to maintain SIMIT operability.

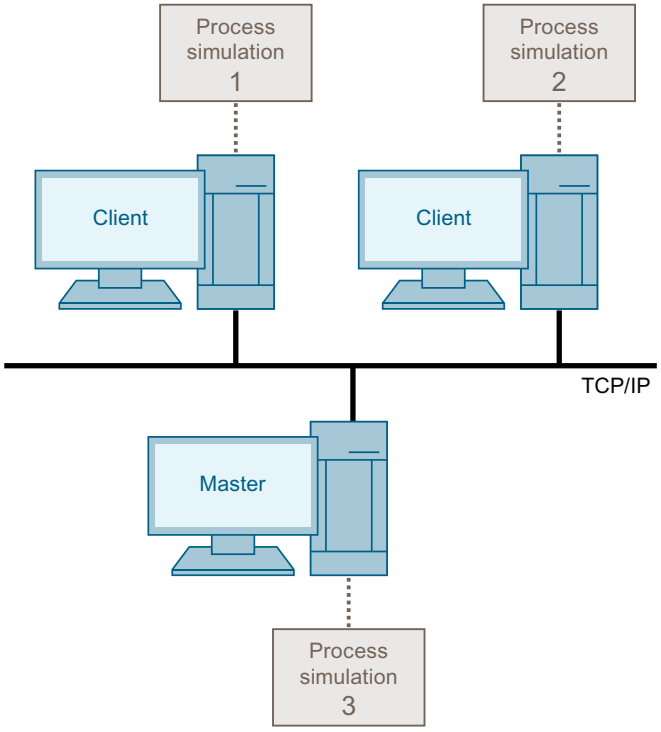
These timeout times are set so that as a general rule they are not exceeded. If you still receive a message that a timeout has been exceeded, check first to see whether it is due to a SIMIT error. If not, you can increase the timeout times in the Windows registry. You can find the corresponding keys under "HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\SIMIT\8.0\Timeout" or "HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Siemens\SIMIT\8.0\Timeout" on 64-bit operating systems.

1.7 Distributed simulation

1.7.1 Method of operation of the distributed simulation

Method of operation

The complexity of the simulation project is limited by the computing power of the simulation PC. To realize complex simulation projects with SIMIT, distribute a simulation project to up to 16 client PCs. You control the simulation centrally from the master PC.



Typical scenarios for the use of distributed simulation are, for example,

- Plant components act mostly independent of one another
- Little or no time-critical data exchange between the plant components
- Power plant with multiple blocks
- Manufacturing process with multiple lines

1.7.2 Configuration of a distributed simulation

Example for configuration of a distributed simulation

Based on the plant structure, a simulation project is divided up into multiple subprojects. To do so, the simulation project is copied to the client PCs involved and stored in the same local path as on the master PC. The simulation project is customized on each client PC. Each simulation project can run independently.

Note

The same version of SIMIT must be installed on all PCs.

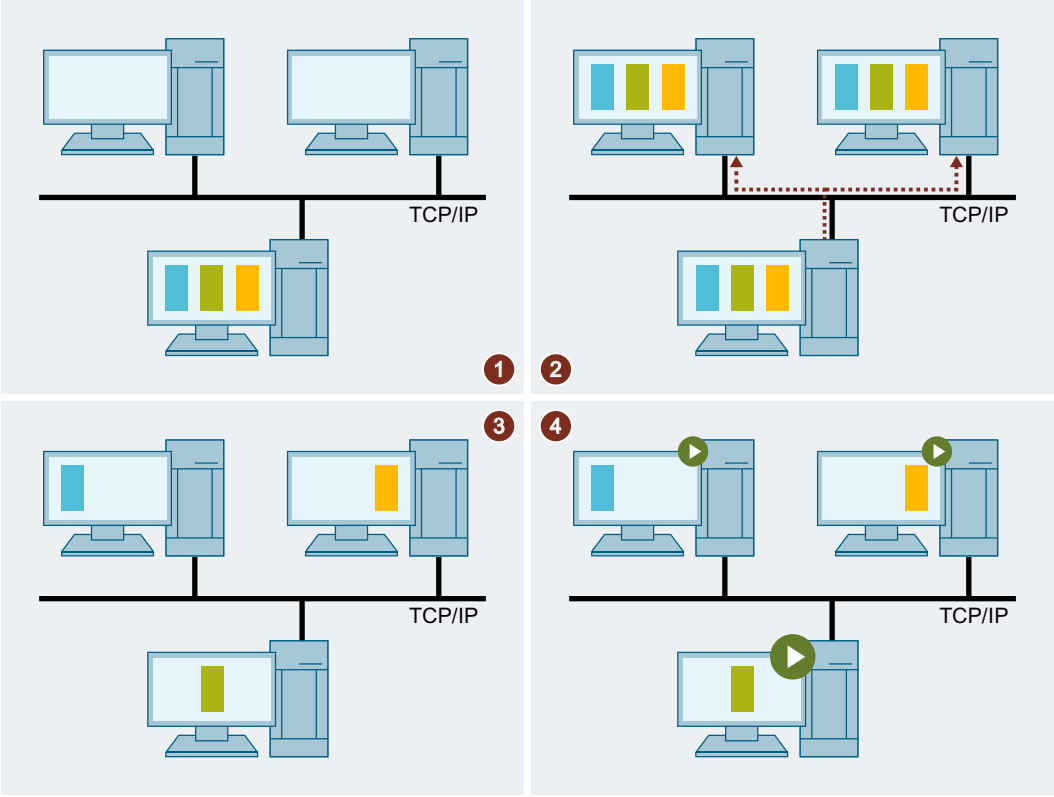
Note

Power saving options

After installation of SIMIT SP or SIMIT VC, disable the power saving options of the network adapters and the system on all computers. This will prevent shutdown of network and system functions by the operating system in simulation mode.

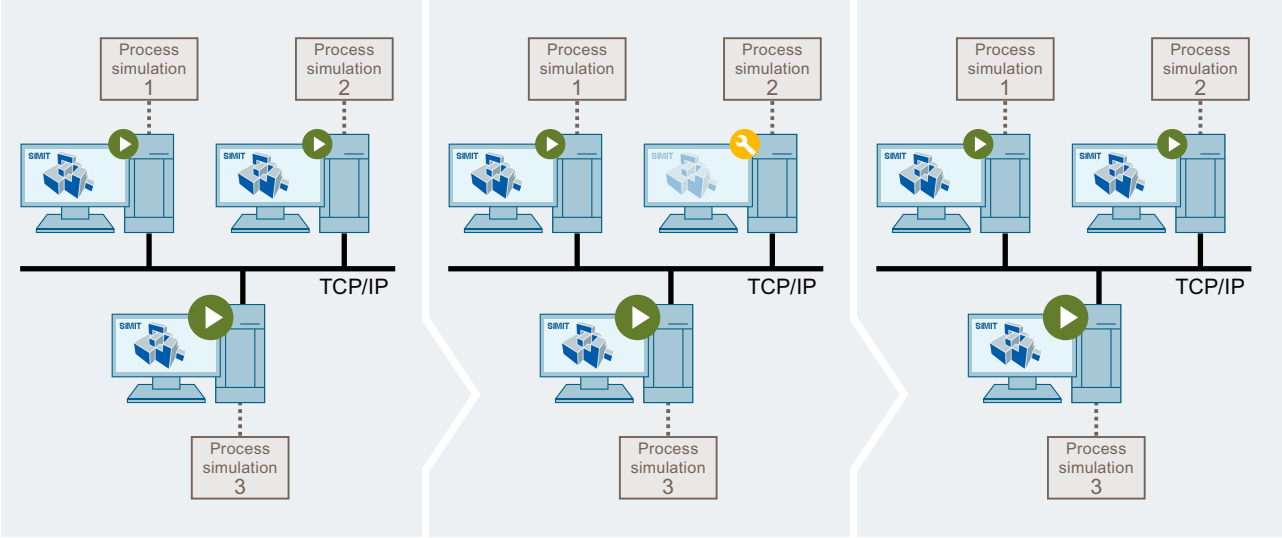
Detailed information is available in the documentation of your operating system.

The figure below shows the basic approach:



① Create simulation project on master PC
② Copy simulation project to client PCs
③ Configure simulation and test for each client PC and master PC
④ Start simulation on master PC

Sequence of a distributed simulation



① When you start the simulation on the master PC, you also start the partial simulations on the client PCs.

② To change a subproject on a client PC, disable the remote control of the client PC by the master PC. Then change the model on the client PC.

③ When you subsequently restore the remote control of the client PC by the master PC, all partial simulations are once again synchronous again.

See also

Changing distributed simulation on a client PC (Page 53)

Start distributed simulation (Page 52)

1.7.3 Data exchange in a distributed simulation

Distributed simulation with data exchange between client PCs and master PC

If your simulation cannot be completely subdivided into independent partial simulations, use the distributed simulation with data exchange. Divide your project into subprojects so that the partial simulations exchange as little data as possible.

Use the OPC DA coupling, for example, for the data exchange. In the example below, the master PC of the distributed simulation is also the OPC DA server. The client PCs of the distributed simulation are the OPC DA clients.

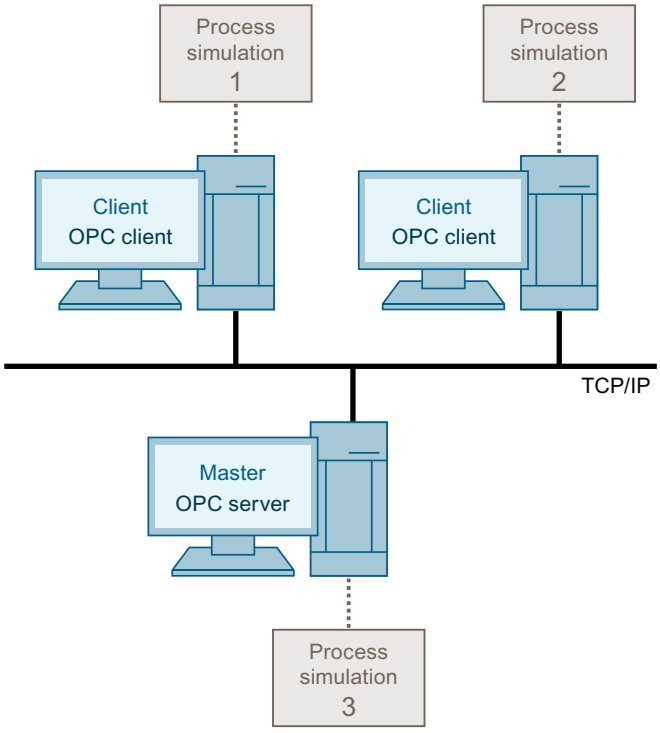


Figure 1-1 Distributed simulation with data exchange between client PCs and master PC

1.7.4 Setup firewall

Introduction

Set the firewall to ensure that SIMIT SP can establish a connection to the following systems:

- Further SIMIT SP instances in a distributed simulation
- Distributed Virtual Controllers
- Third-party systems which establish a connection to SIMIT SP using a simulation controller (RCI)

NOTICE
Read the security notes included in the foreword Changes to the firewall may influence the security of your system! Contact, if necessary, your system administrator before performing the steps described in the following.

Requirement

- SIMIT has been installed.
- All nodes are located within the same network.

Procedure

Proceed as follows to set up the firewall:

1. Open the "Windows Defender Firewall with extended safety", for example, by entering "wf.msc" in the system prompt and confirming with "Enter".
2. Click on “Incoming rules”.
3. Double-click "SIMIT-CS Manager".
   The “Properties of SIMIT CS Manager" windows opens.
4. Select the “Area” tab.
5. Under “Remote IP address”, select “Arbitrary IP address”. Alternatively, add the IP addresses of the systems involved to the list of approved remote IP addresses.
6. Confirm with "OK".

Result

The firewall has been set such that SIMIT SP can set up a connection to other systems.

See also

Preface (Page 3)

1.7.5 Create screen link "SIMIT Client"

Introduction

To start SIMIT in client mode, create a screen link with a corresponding transfer parameter.

Syntax of the transfer parameter: /[M|m]:<IP address>:<Port>

- /M: The operator of the client PC can disable the remote control.
- /m: The operator of the client PC cannot disable the remote control.
- <IP address>: IP address of the master PC
- <Port>: Port number of the master PC, default value: 50800.
  If an administrator has changed the default port, you can locate the current port on the master PC in the registry under [HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node \Siemens\SIMIT\8.0] in the "uri" key.

Example for a complete transfer parameter:

"C:\Program Files (x86)\Siemens\Automation\SIMIT\SIMIT\SIMIT SF\bin\SIMIT.exe" /
M:192.168.0.1:50800.

Requirement

- SIMIT is installed.
- The firewall configuration of the master PC permits incoming connections at the port number.

Procedure

To create a "SIMIT Client" screen link, follow these steps:

1. Copy the link "SIMIT SP" to the screen.
2. Add the transfer parameter in the properties of the copied link.
3. Rename the link to "SIMIT Client".

Result

The "SIMIT Client" screen link has been created.

1.7.6 Start distributed simulation

Requirement

- The master PC and the client PCs are located in the same network.
- A project with the same name is stored on the master PC and the client PCs under the same local path.
- The screen link (Page 51) "SIMIT Client" has been set up on all client PCs.

Procedure

To start a distributed simulation, follow these steps:

1. Start SIMIT on the master PC.
2. On the client PCs, start SIMIT in client mode.
3. Open the project on the master PC.
4. Start the simulation on the master PC.

Result

The simulation runs on the master PC and on the client PCs.

See also

Configuration of a distributed simulation (Page 47)

1.7.7 Changing distributed simulation on a client PC

Introduction

No model changes are possible on the client PC during an ongoing distributed simulation. To change the model, interrupt the connection between the client PC and the master PC.

Requirement

- A distributed simulation runs on the master PC and on the client PCs.
- The client PC was started with the transfer parameter "/M".

Procedure

To change a distributed simulation on the client PC, follow these steps:

1. Deactivate the "Remote control active" option in the project properties.
   The connection to the master PC is disconnected. The simulation on the client PC initially continues.
2. Change the model.
3. Start the simulation with your changes.
4. When the simulation can be run, bring the simulation on the client PC into the same state as the simulation on the master PC.
5. Activate the "Remote control active" option in the project properties.

Result

The client PC once again runs under the control of the master PC.

See also

Configuration of a distributed simulation (Page 47)

Couplings

2

2.1 Coupling concept

2.1.1 Couplings in the SIMIT architecture

Introduction

A coupling is an interface between SIMIT and a coupling partner, for example, an automation system. A coupling has the following tasks:

- Signal exchange with the coupling partner
  The input/output signals (I/O signals) of the coupling partner are exchanged with SIMIT over the coupling.
- Coordination of signal exchange between SIMIT and the coupling partner

Types of coupling

There are three different types of coupling in SIMIT:

- Coupling with a real SIMATIC controller
- Coupling with an emulated SIMATIC controller
- Coupling with an external partner

	Coupling with a real SIMATIC controller		**Coupling with an emulated SIMATIC controller**			**Coupling with an external partner**
	SIMIT Unit	**PRODAVE**	**Virtual controller**	**PLCSIM**	**PLCSIM Advanced**	**–**
Simulation of bus behavior	Yes	No	No			No
Simulation of isochronous mode	No		No		Yes	No[1]
Access to the process image	Yes		Yes			No
Use of hardware configuration data from STEP 7	Yes	No	Yes	No	No	No
Import and export of signals	Yes		Yes			Yes[2]

[1] The MCD coupling supports isochronous mode.

[2] Depending on the external coupling partner

Each coupling has its own configurator for creating and editing the coupling.

Note

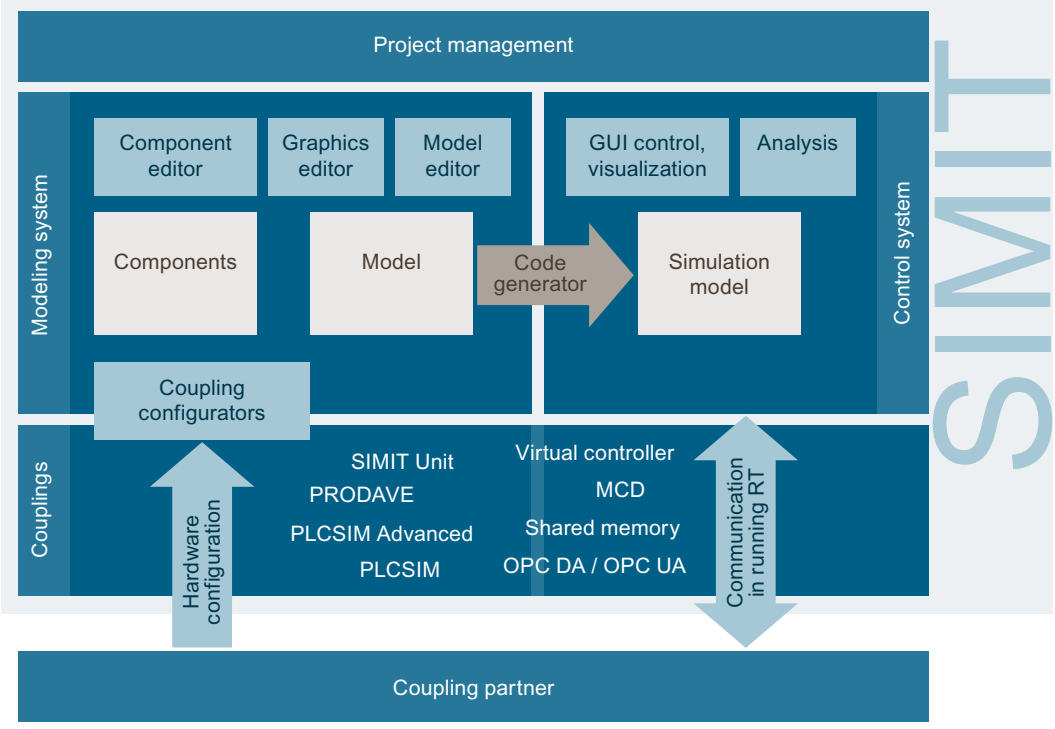
A maximum of 16 external couplings can be used in a SIMIT project, regardless of the coupling type.

Coupling can be located in no more than 7 different time slices in a SIMIT project.

Couplings in the architecture of SIMIT

Each simulation system consists of two components:

- Modeling system
  In the modeling system you create the simulation model and configure the interfaces to the coupling partners.
- Control system
  You run the simulation model in the control system. To do so, the simulation model communicates with the coupling partners.



A coupling provides the connection between the simulation model run by the control system and the coupling partner.

- The interface to the simulation model
  This interface is the same for all couplings. The simulation model therefore does not depend on the type of coupling used. Couplings can therefore be interchanged within the simulation model without having to be modified.
- The interface to the coupling partner
  The interface must be configured once for each coupling. The interface is different for each coupling.

Configuration of couplings

For signal exchange between the simulation model and an automation system, the coupling must contain the following information:

- The information required to establish the connection
- Signals to be exchanged

This information is saved for each coupling with a configurator.

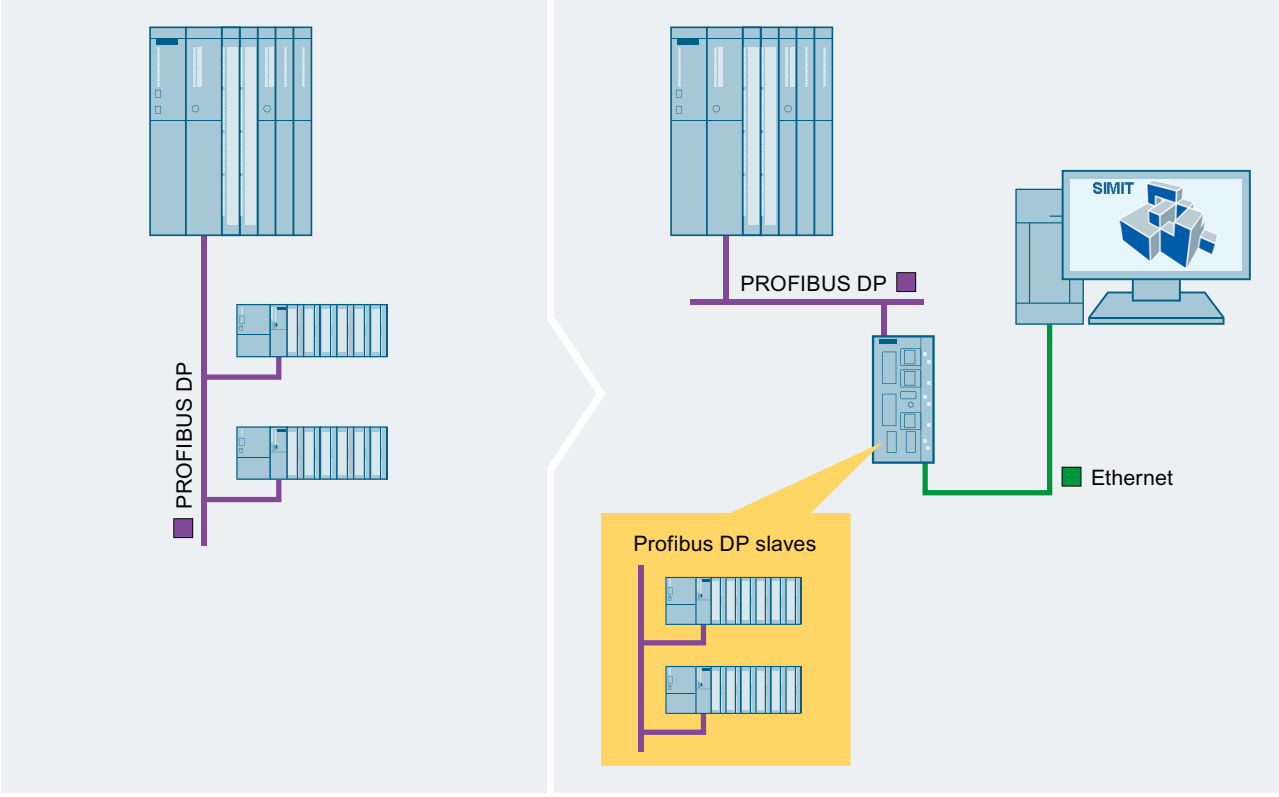
2.1.2 Couplings to SIMATIC PLCs

Introduction

SIMIT provides couplings with a SIMATIC controller such as S7-400 over PROFIBUS DP, PROFINET IO or PRODAVE and a coupling to PLCSIM / PLCSIM Advanced and to SIMIT VC. The data required for the configuration can be taken from the SIMATIC projects.

Coupling with a SIMATIC controller

You couple a SIMATIC controller with a PROFIBUS DP or PROFINET IO field bus directly at the field bus level with SIMIT:



"SIMIT Units" that simulate the field devices are used in SIMIT to do this. Such an SIMIT Unit is connected to the PROFIBUS DP master or PROFINET IO controller of the control system in the simplest configuration. The controller then communicates with this SIMIT Unit and with SIMIT in the same way as in a real plant with the field devices. The SIMIT interface to the SIMATIC controller is in this case the field bus cable.

You can find additional information on this under SIMIT Unit (Page 65).

Coupling with Virtual Controller

A virtual controller simulates a station with process image input and output. You can find additional information on this under Virtual Controller (Page 122).

Coupling with PLCSIM Advanced

PLCSIM Advanced coupling of SIMIT enables the dynamic exchange of data between PLCSIM Advanced and SIMIT. PLCSIM Advanced and SIMIT must be installed on the same computer. You can find additional information on this under PLCSIM Advanced coupling (Page 157).

Coupling with PLCSIM

The SIMIT PLCSIM coupling enables the dynamic exchange of data between PLCSIM and SIMIT. PLCSIM and SIMIT must be installed on the same computer. You can find additional information on this under PLCSIM coupling (Page 163).

Coupling with PRODAVE

An alternative to coupling SIMIT to a SIMATIC controller over a field bus is coupling over the PRODAVE interface. The physical connection between the controller and the SIMIT PC is in this case over one of the following interfaces:

- Serial interface
- USB
- MPI interface card
- Ethernet

This coupling does not offer such high performance as the field bus coupling due to the PRODAVE interface.

You can find additional information on this under PRODAVE coupling (Page 200).

Note

IP address space

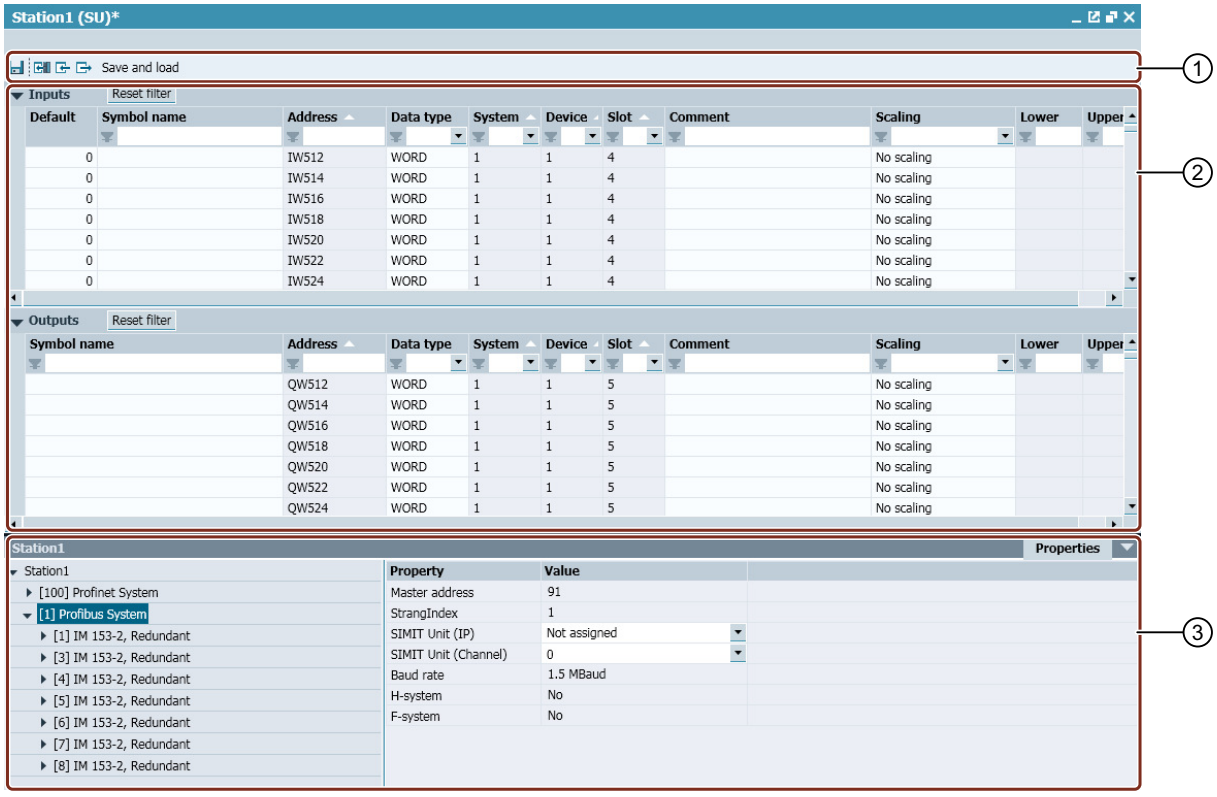
SIMIT only supports PCS 7 projects with an IP V4 address space.

2.2 Coupling editor

You execute the following tasks in the Coupling Editor:

- Configuring the properties of a coupling
- Configuring the properties of signals

The coupling editor is specific to each type of coupling, but always has the structure shown in the figure below:



① Menu bar

The content of the menu bar depends on the coupling type. By default, the menu bar includes the commands for importing and exporting signals. For further information, please refer to: Importing and exporting signals (Page 221)

② Input and output signals

Lists the signals with their associated parameters and properties separately by "Inputs" and "Outputs":

- You connect the signals under "Inputs" to the output connectors in the simulation model.
- You connect the signals under "Outputs" to the input connectors in the simulation model.

You can find information on editing signals in the coupling editor in: Signal basics (Page 210).

③ Property view

In the property view you edit the properties of a signal or those of the coupling you selected in the project tree.

The properties that are displayed depend on the type of coupling.

Double-click on a coupling in the project tree to open the coupling editor.

2.3 Activation and deactivation of couplings

2.3.1 Activation and deactivation of couplings

All created couplings can be activated and deactivated. This function simplifies the processing of the simulation models.

For example, if two signals with the same name exist in two different couplings, the deactivated coupling is not taken into consideration during the consistency check. The signal of the deactivated coupling is therefore not simulated.

The signals of deactivated couplings are not displayed in the "Signals" task card.

The deactivated couplings are still taken into consideration by the "Find & replace (Page 323)" and "Analysis (Page 329)" functions.

The maximum number of couplings that can be created in a simulation project remains unchanged.

Two couplings with the same name are still only permitted to be created in different folders.

A deactivated coupling is marked with the following symbol.



2.3.2 Activating couplings

Requirement

A coupling is deactivated in the project navigation.

Procedure

1. Select the coupling you want to activate.
2. Select "Activate" in the shortcut menu.

Result

The selected coupling has been activated.

2.3.3 Deactivating couplings

Requirement

A coupling is created and configured in the project navigation.

Procedure

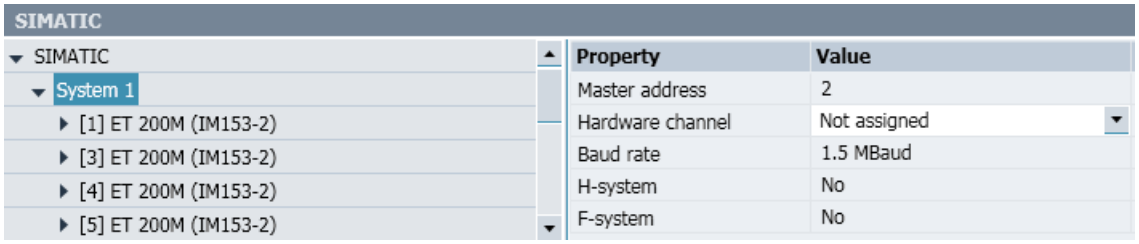
1. Select the coupling you want to deactivate.
2. Select "Deactivate" in the shortcut menu.

Result

The selected coupling has been deactivated.

2.3.4 Revoking coupling assignments

If you have configured a coupling in your simulation project but want to start simulation without it, set the hardware channel in the coupling property view to "Not assigned":



The next time the simulation is started, no signal exchange of this coupling occurs. All values for the signals in this coupling are set to "0".

This function is only available with the following types of coupling:

- SIMIT Unit
- PLCSIM
- PRODAVE
- OPC client

2.4 HWCNExport

2.4.1 HWCNExport introduction

Introduction

HWCNExport is included in the standard scope of delivery of SIMIT.

The "HWCNExport" application creates an HWCNExport file from a STEP 7/PCS 7 project or TIA project.

Note

The "HWCNExport" application can be run without additional installation.

You can find the application in the folder that you reach via the Start menu "Start > All Programs > SIMIT > Tools".

If you want to use this application on a PC without a SIMIT installation, copy the complete folder to any location on the PC and start the "HWCNExport".

The following couplings support import from an HWCNExport file:

- Virtual controller
- PLCSIM Advanced in a compatible version, see Compatibility tool (www.siemens.com/kompatool)

See also

"Virtual controller import" dialog box (Page 985)

"PLCSIM Advanced import" dialog box (Page 984)

2.4.2 Creating a HWCNExport file from STEP 7 projects

Requirement

- STEP 7 is installed.

  Note

  For STEP 7 V5.6, the version STEP 7 V5.6 SP1 + HF1 is required.

- .NET Framework 3.5 or higher is installed.
- The "HWCNExport" application is available.

Procedure

1. Launch the "HWCNExport" application.
2. Select the "STEP 7 V5.x" tab.
3. Select the desired STEP 7/PCS 7 project type.
4. Select the desired STEP 7/PCS 7 project.
5. Select a folder where the HWCNExport file will be stored.
6. Click "Create XML".

Result

The HWCNExport file in "*.XML" format has been created and can be imported into a SIMIT project.

Importing a STEP 7/PCS 7 project (Page 142)

2.4.3 Creating an HWCNExport file from TIA projects

Requirement

- TIA Portal is installed in a compatible version. See Compatibility tool (www.siemens.com/kompatool).
- TIA Portal Openness is installed in a compatible version. See Compatibility tool (www.siemens.com/kompatool).
- .NET Framework 3.5 or higher is installed.
- The "HWCNExport" application is available.

Procedure

1. Launch the "HWCNExport" application.
2. Select the "STEP 7 (TIA Portal)" tab.
3. Select the desired STEP 7 project.
4. Select a folder where the HWCNExport file will be stored.
5. Click "Create XML".

Result

The HWCNExport file in "*.XML" format has been created and can be imported into a SIMIT project.

Importing hardware configuration to station (Page 158)

2.5 SIMIT Unit

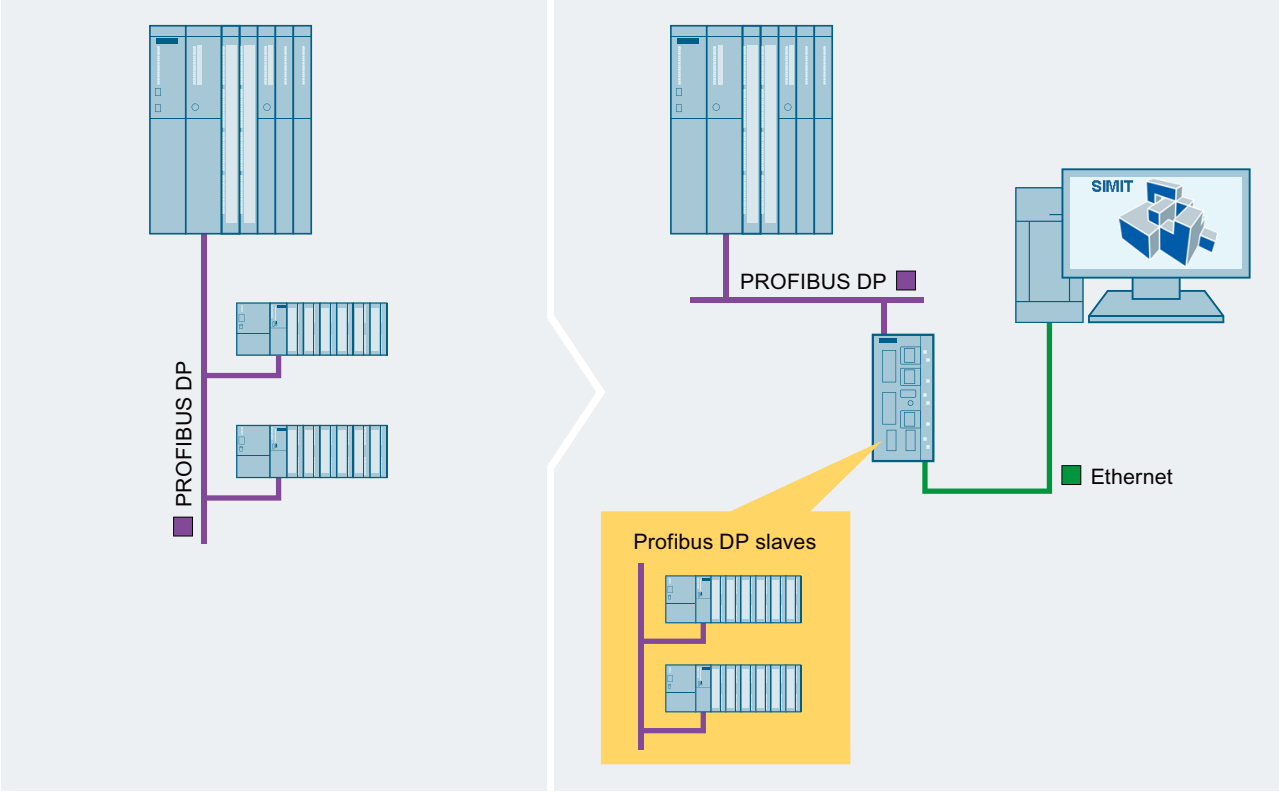
2.5.1 Operating principle of the SIMIT Unit coupling

Function

SIMIT communicates with one or more bus systems over the "SIMIT Unit" coupling:

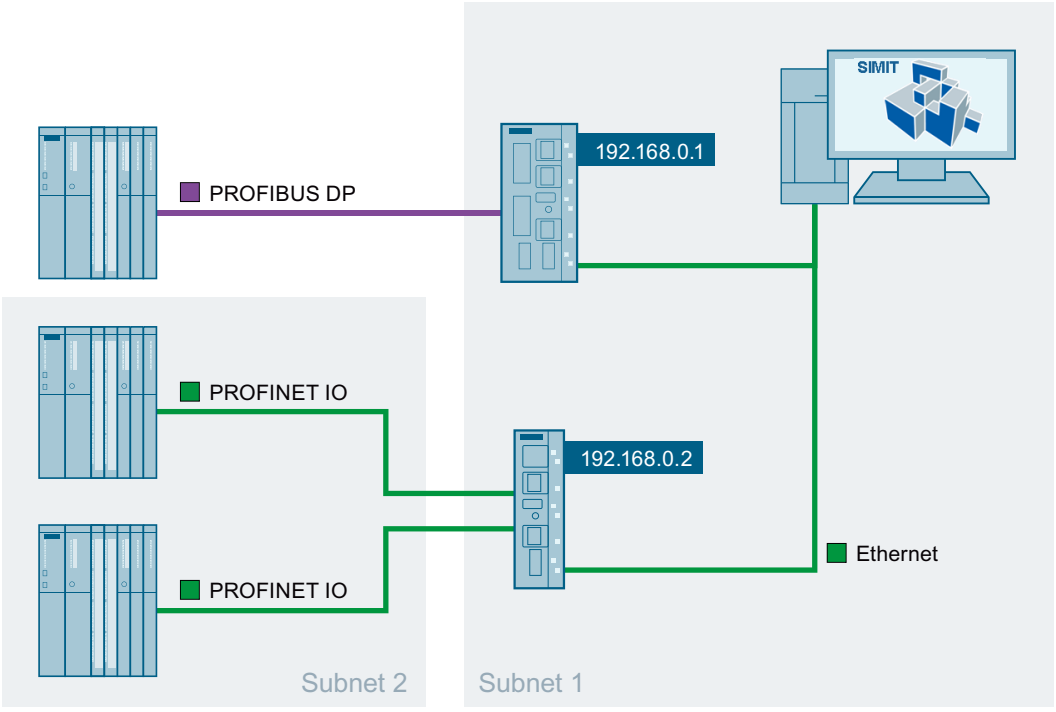
- PROFINET IO controller
- PROFIBUS DP master

The "SIMIT Unit" is used as a link for communication between the controller and SIMIT. The SIMIT Unit maps the behavior of devices at the bus and enables the exchange of data between the controller and SIMIT.



SIMIT Unit connection to SIMIT

The SIMIT Unit communicates with a SIMIT PC over Ethernet. SIMIT PC and all connected SIMIT Units must be located in one shared subnet. The devices simulated on PROFINET must be located in a different subnet. You can find more information in the section "Supported PROFINET IO configurations (Page 81)".



Additional information

You can find more information on the SIMIT Unit on the Internet (https://support.industry.siemens.com/cs/ww/en/view/109746192).

See also

Configuring a SIMIT Unit (Page 94)

Supported PROFIBUS DP configurations (Page 67)

2.5.2 Supported PROFIBUS DP configurations

Introduction

Typical PROFIBUS DP configurations that are supported by SIMIT are listed below. In the left of the figure (see below), you can see the configuration of the automation system and on the right the layout with SIMIT. The following examples of configurations with non-redundant controllers are shown:

- A controller with a PROFIBUS DP line
- A controller with two PROFIBUS DP lines
- Two controllers, each with a PROFIBUS DP line

The following examples of redundant configurations are provided:

- One redundant controller (two CPUs present) with one redundant PROFIBUS DP line
- One redundant controller with Y-Link
- One redundant controller (two CPUs present) with two redundant PROFIBUS DP lines
- One redundant controller (only one CPU present) with one redundant PROFIBUS DP line
- Two redundant controllers (four CPUs present) with one redundant PROFIBUS DP line each
- Two redundant controllers (one CPU present for Controller 1 and one CPU for Controller 2) with one redundant PROFIBUS DP line each
- Two redundant controllers, each with a PROFIBUS DP line configured at one end only

For all configurations, a mixed configuration of real and simulated PROFIBUS DP slaves is possible.

Bus couplers can also be used, either as DP/DP couplers, as DP/PA couplers, or in redundant configurations as a Y-Link.

Fail-safe I/Os can also be simulated in redundant H/F systems.

Examples with non-redundant controllers

The table shows for different configurations how the ports PB1 and PB2 of the SIMIT Unit should be connected. The table shows whether port PB2 can be connected.

Configuration	A controller with a PROFIBUS DP line	A controller with two PROFIBUS DP lines	Two controllers, each with a PROFIBUS DP line
Connect port PB1 with	✓ Line 1 from the controller	✓ Line 1 from the controller	✓ Line 1 from controller 1
Connect port PB2 with	✗ Cannot be connected	✓ Line 2 from the controller	✓ Line 1 from controller 2
Number of required SIMIT Units	1	1	1

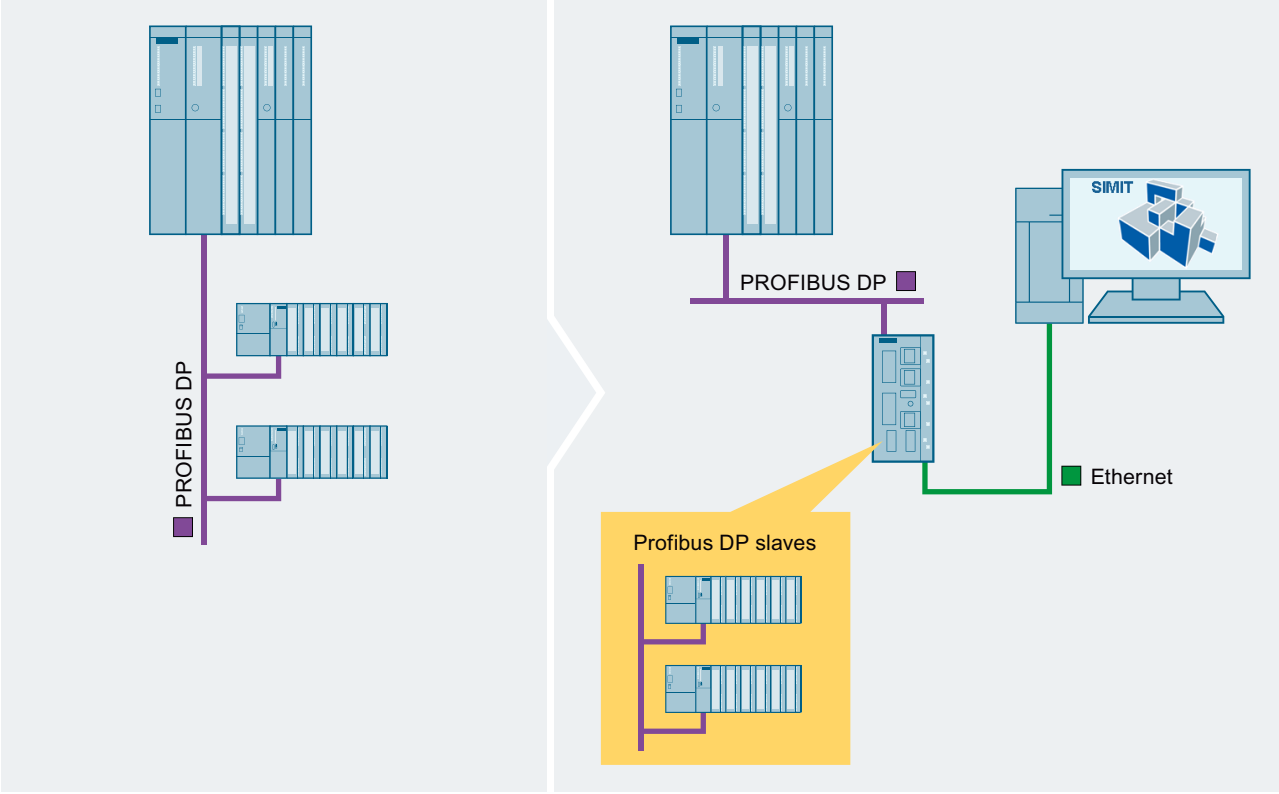
Examples with redundant controllers

The table shows for different configurations how the ports PB1 and PB2 of the SIMIT Unit should be connected. The table shows whether port PB2 can be connected and the configurations in which a second SMIT Unit needs to be used.

Configuration	**One redundant controller (two CPUs present) with one redundant PROFIBUS DP line**	**One redundant controller (two CPUs present) with two redundant PROFIBUS DP lines**	**One redundant controller (only one CPU present) with one redundant PROFIBUS DP line**	**Two redundant controllers (four CPUs present) with one redundant PROFIBUS DP line each**	**Two redundant controllers (one CPU present for Controller 1 and one CPU for Controller 2) with one redundant PROFIBUS DP line each**
Connect port PB1 with	✓ Left line from the redundant controller	✓ Left line 1 from the redundant controller	✓ Left line from the redundant controller	✓ Left line 1 from redundant controller 1	✓ Left line 1 from redundant controller 1
Connect port PB2 with	✓ Right line of the redundant controller	✓ Right line 1 from the redundant controller	✗ Cannot be connected	✓ Right line 1 from redundant controller 1	✗ Cannot be connected
		✗ Not possible: Connect with left or right line 2 from the redundant controller. Solution: Use second SIMIT Unit PB		✗ Not possible: Connect with left or right line 1 from redundant controller 2. Solution: Use second SIMIT Unit PB	✗ Not possible: Connect with left line 1 from redundant controller 2. Solution: Use second SIMIT Unit PB
Number of required SIMIT Units	1	2*	1	2*	2*
* Alternative solution: Change the controller configuration in STEP 7 so that no redundancy is used.					

A controller with a PROFIBUS DP line

The controller is connected with the PROFIBUS line to port PB 1 of the SIMIT Unit.

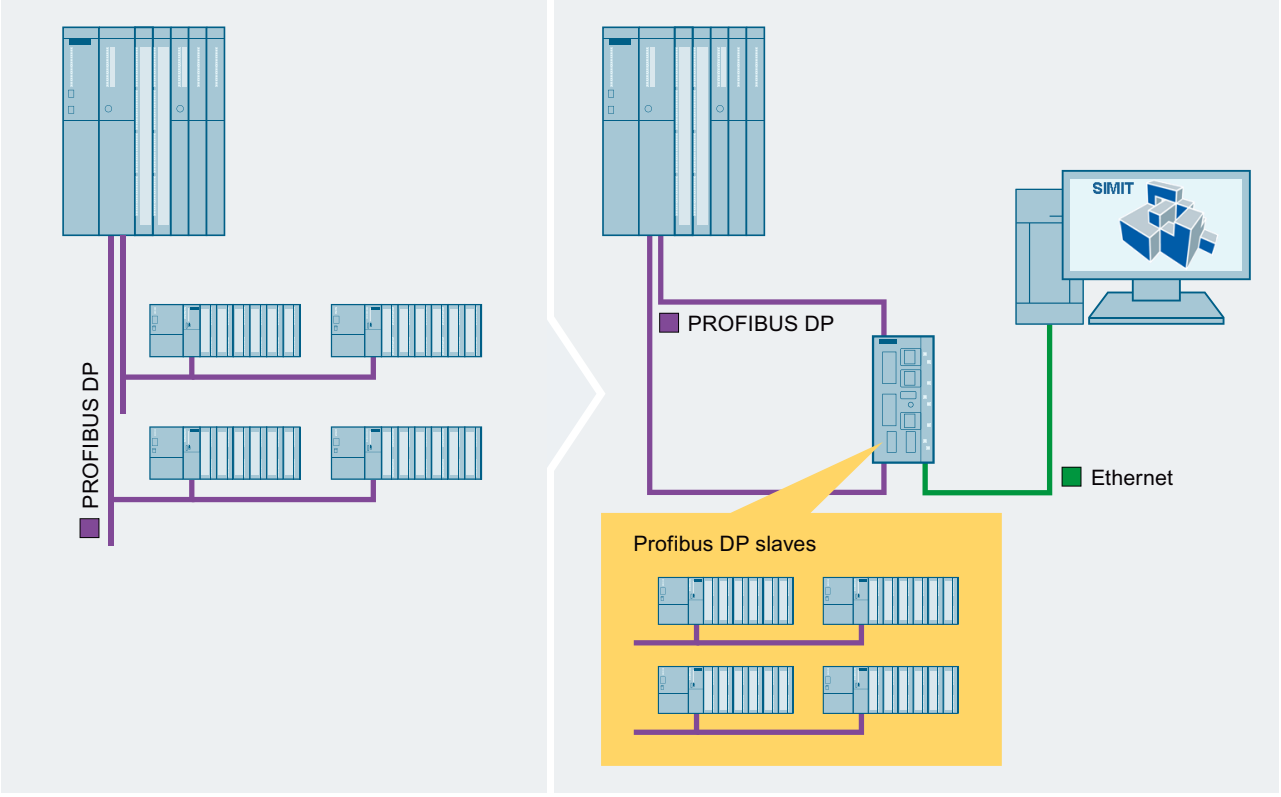


Note

In this configuration, only one channel of a two-channel SIMIT Unit is used.

A controller with two PROFIBUS DP lines

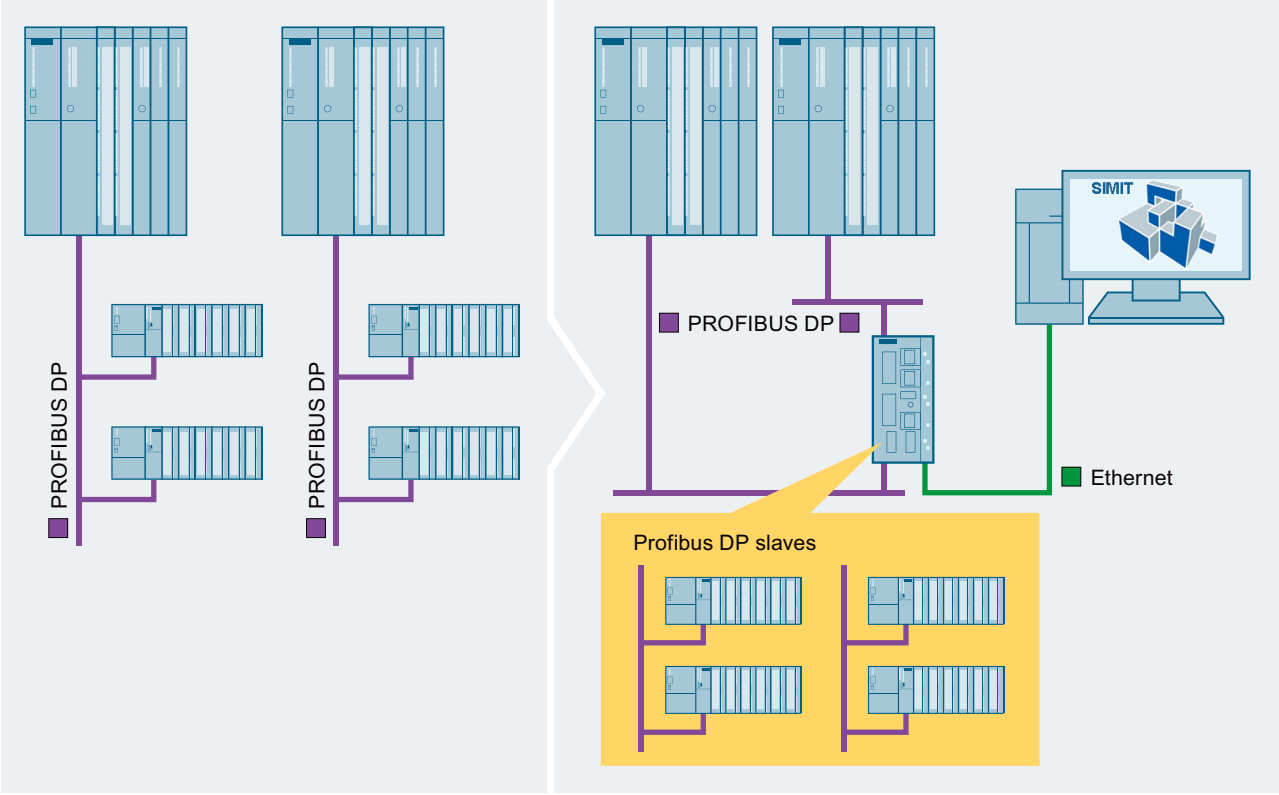
The controller is connected with PROFIBUS line 1 to port PB 1 and with PROFIBUS line 2 to port PB 2 of the SIMIT Unit.



Two controllers, each with a PROFIBUS DP line

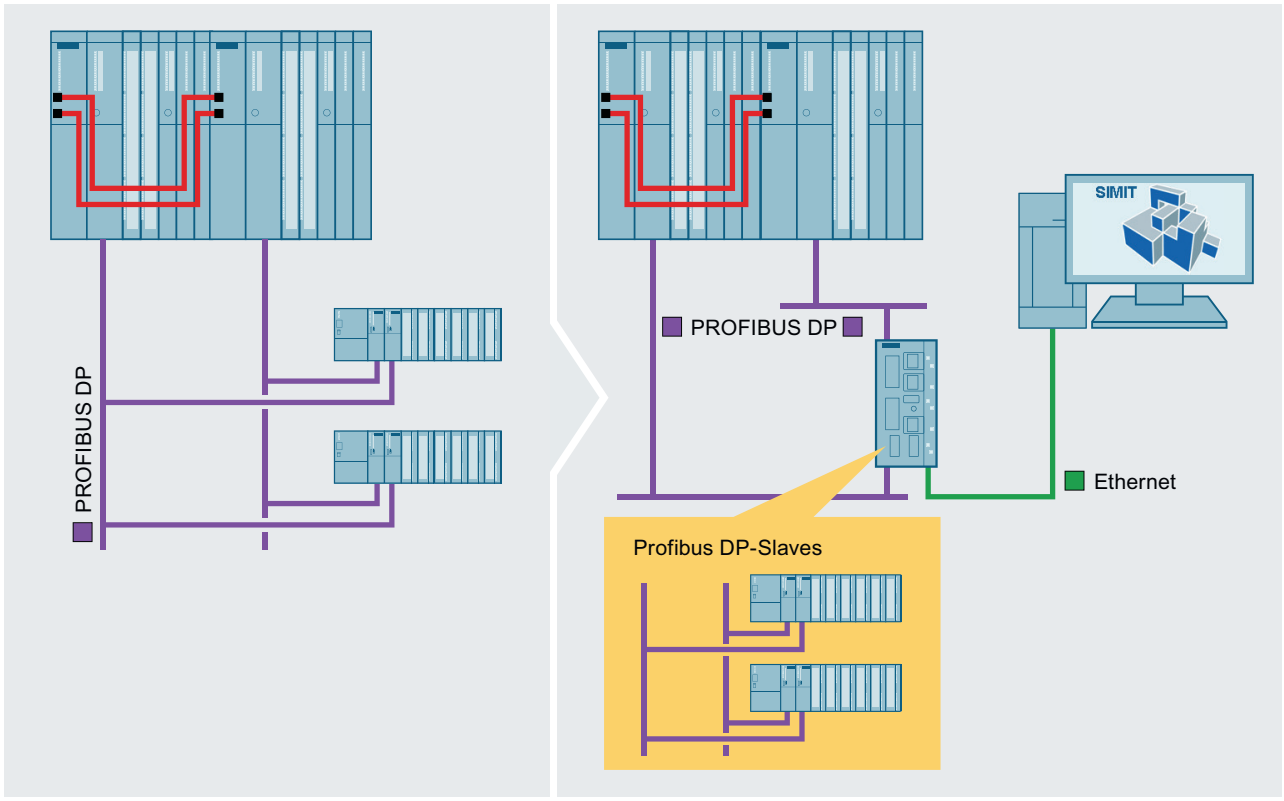
Controller 1 is connected with the associated PROFIBUS line to port PB 1 of the SIMIT Unit.

Controller 2 is connected with the associated PROFIBUS line to port PB 2 of the SIMIT Unit.


PROFIBUS DP
PROFIBUS DP
PROFIBUS DP
SIMIT
Ethernet
Profibus DP slaves

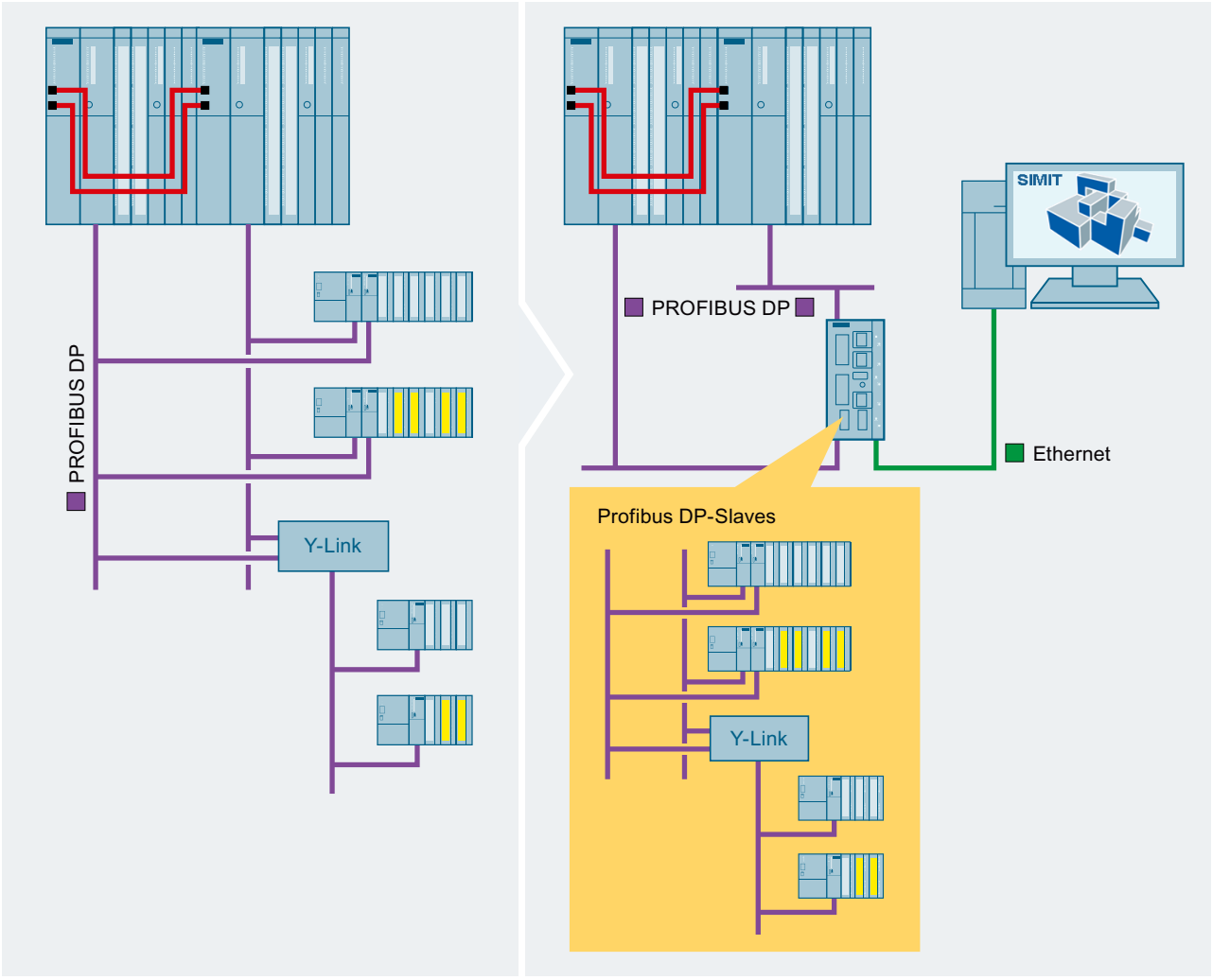
One redundant controller (two CPUs present) with one redundant PROFIBUS DP line

The redundant controller is connected with the left PROFIBUS line to port PB 1 and with the right PROFIBUS line to port PB 2 of the SIMIT Unit.



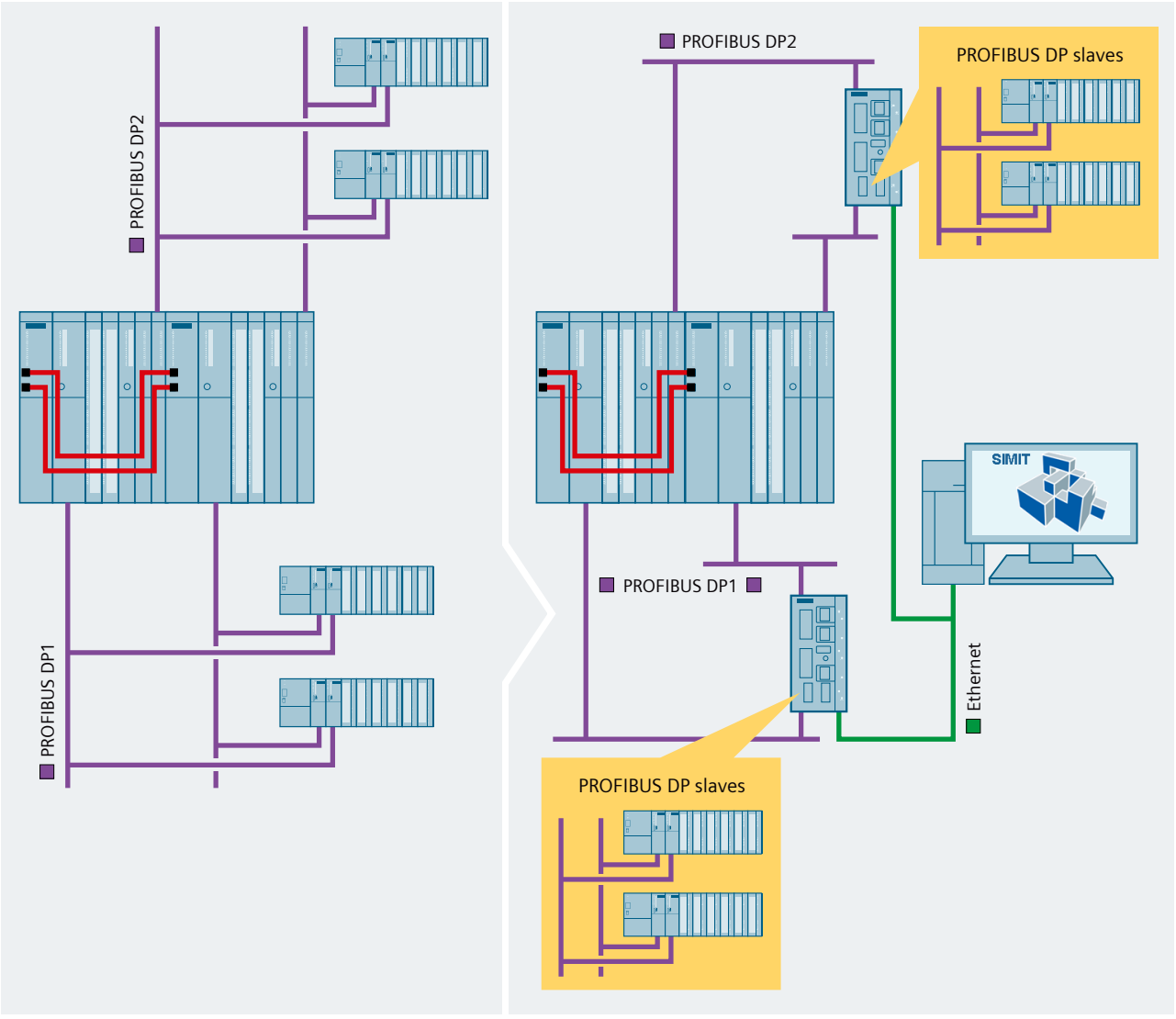
One redundant controller with Y-Link

The redundant controller is connected with the left PROFIBUS line to port PB 1 and with the right PROFIBUS line to port PB 2 of the SIMIT Unit.



One redundant controller (two CPUs present) with two redundant PROFIBUS DP lines

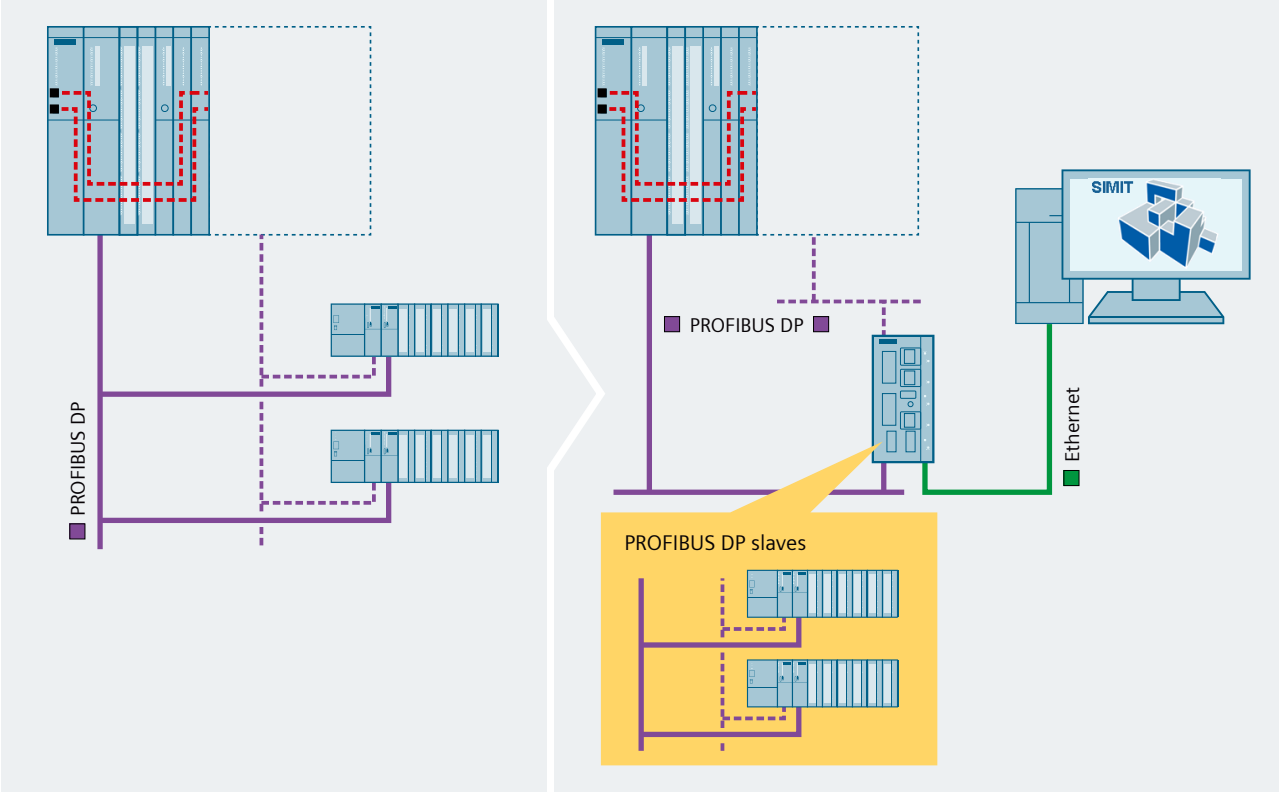
The redundant controller is connected with the left PROFIBUS line to port PB 1 and with the right PROFIBUS line 1 to port PB 2 of the SIMIT Unit. Both ports of the SIMIT Unit are occupied. PROFIBUS line 2 must be connected to a second SIMIT Unit.



One redundant controller (only one CPU present) with one redundant PROFIBUS DP line

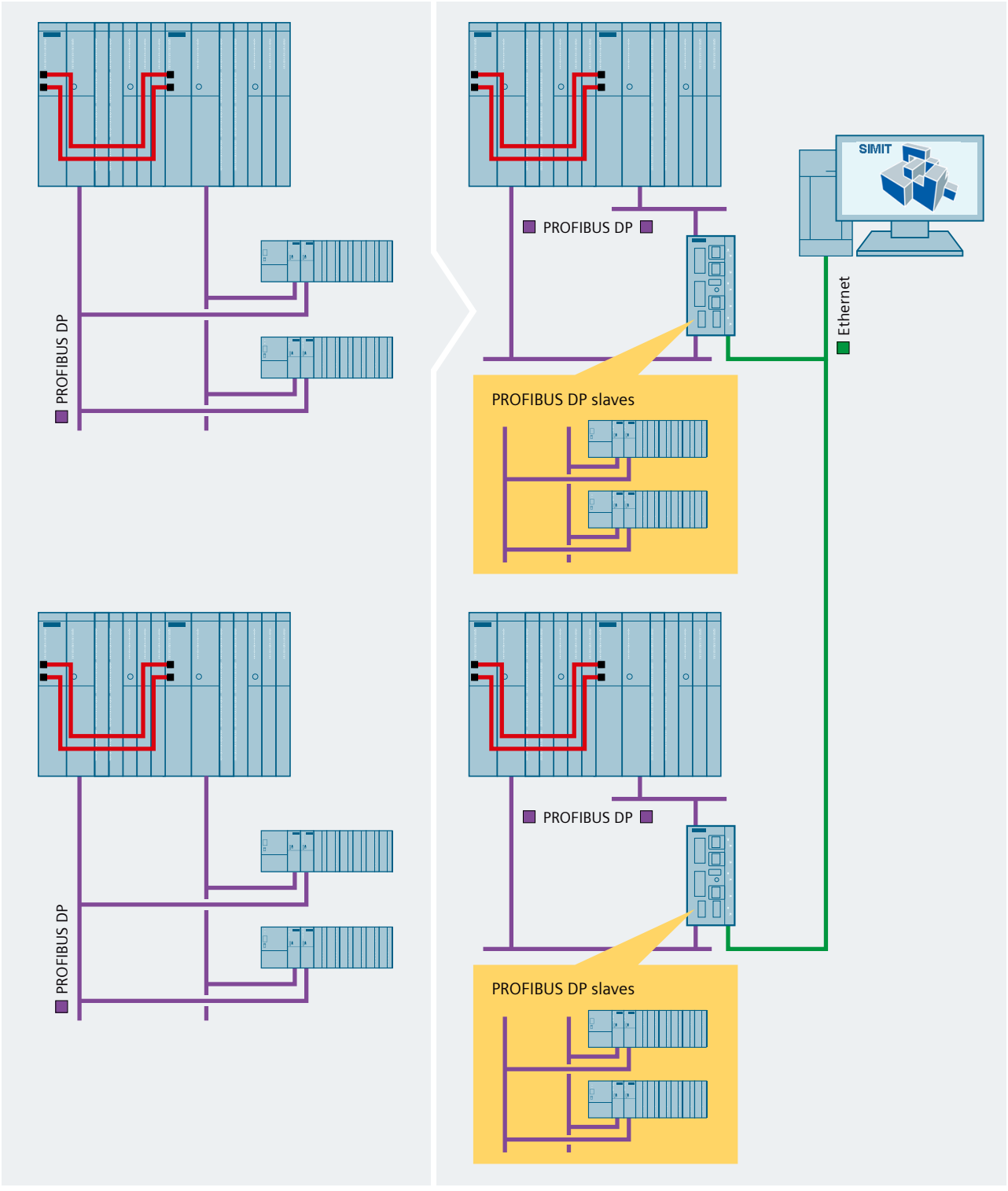
The redundant controller is connected with the left PROFIBUS line to port PB 1 of the SIMIT Unit. Port PB 2 of SIMIT Unit must not be connected.

The second CPU is not physically present, but it is configured in STEP 7.


PROFIBUS DP
PROFIBUS DP
SIMIT
Ethernet
PROFIBUS DP slaves

Two redundant controllers (four CPUs present) with one redundant PROFIBUS DP line each

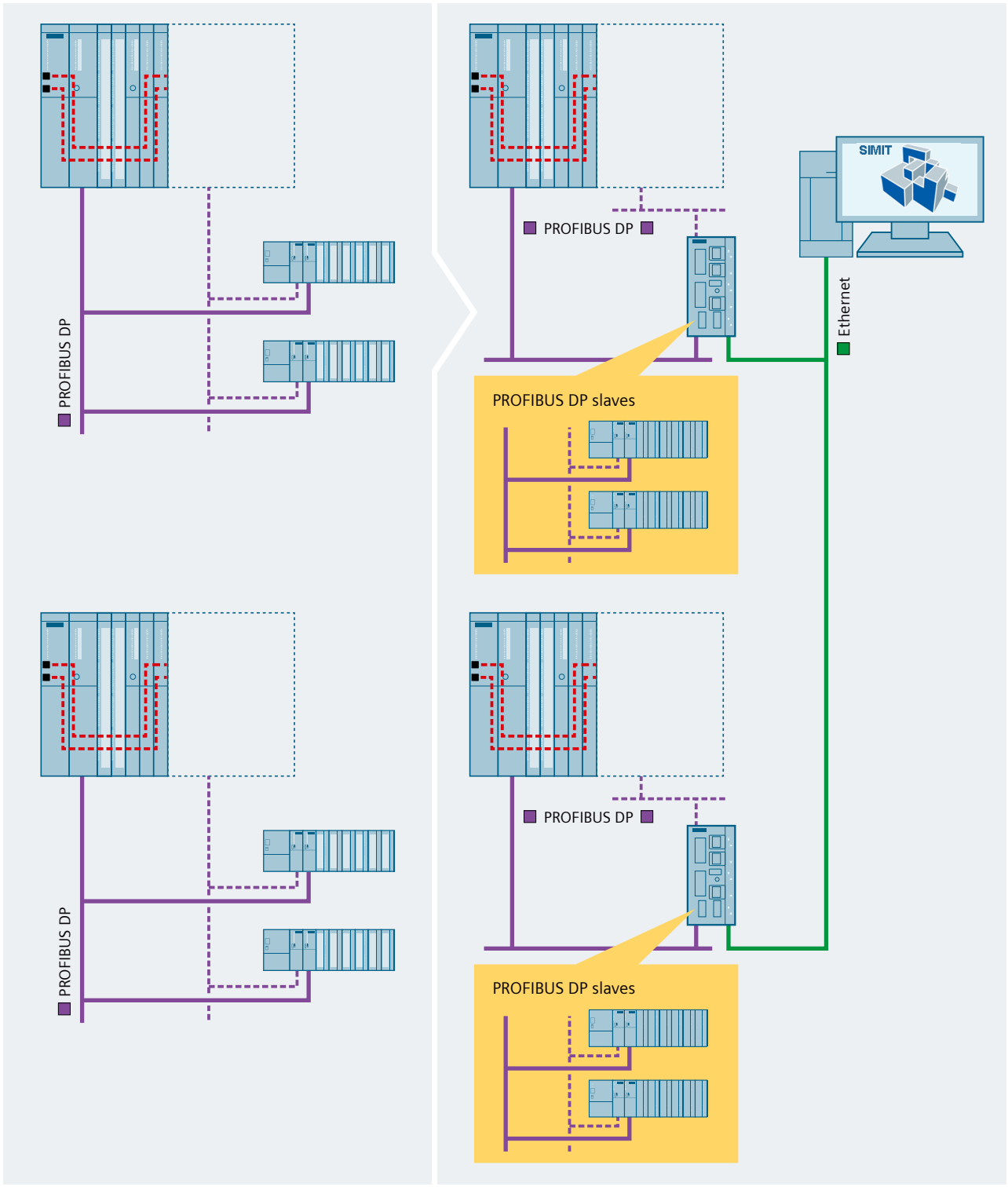
The redundant controller 1 is connected with the left PROFIBUS line 1 to port PB 1 and with the right PROFIBUS line 1 to port PB 2 of the SIMIT Unit. Both ports of the SIMIT Unit are occupied. You need to connect PROFIBUS line 1 of the redundant controller 2 to a second SIMIT Unit.


SIMIT
PROFIBUS DP
PROFIBUS DP slaves
Ethernet
PROFIBUS DP
PROFIBUS DP
PROFIBUS DP slaves
PROFIBUS DP

Two redundant controllers (one CPU present for Controller 1 and one CPU for Controller 2) with one redundant PROFIBUS DP line each

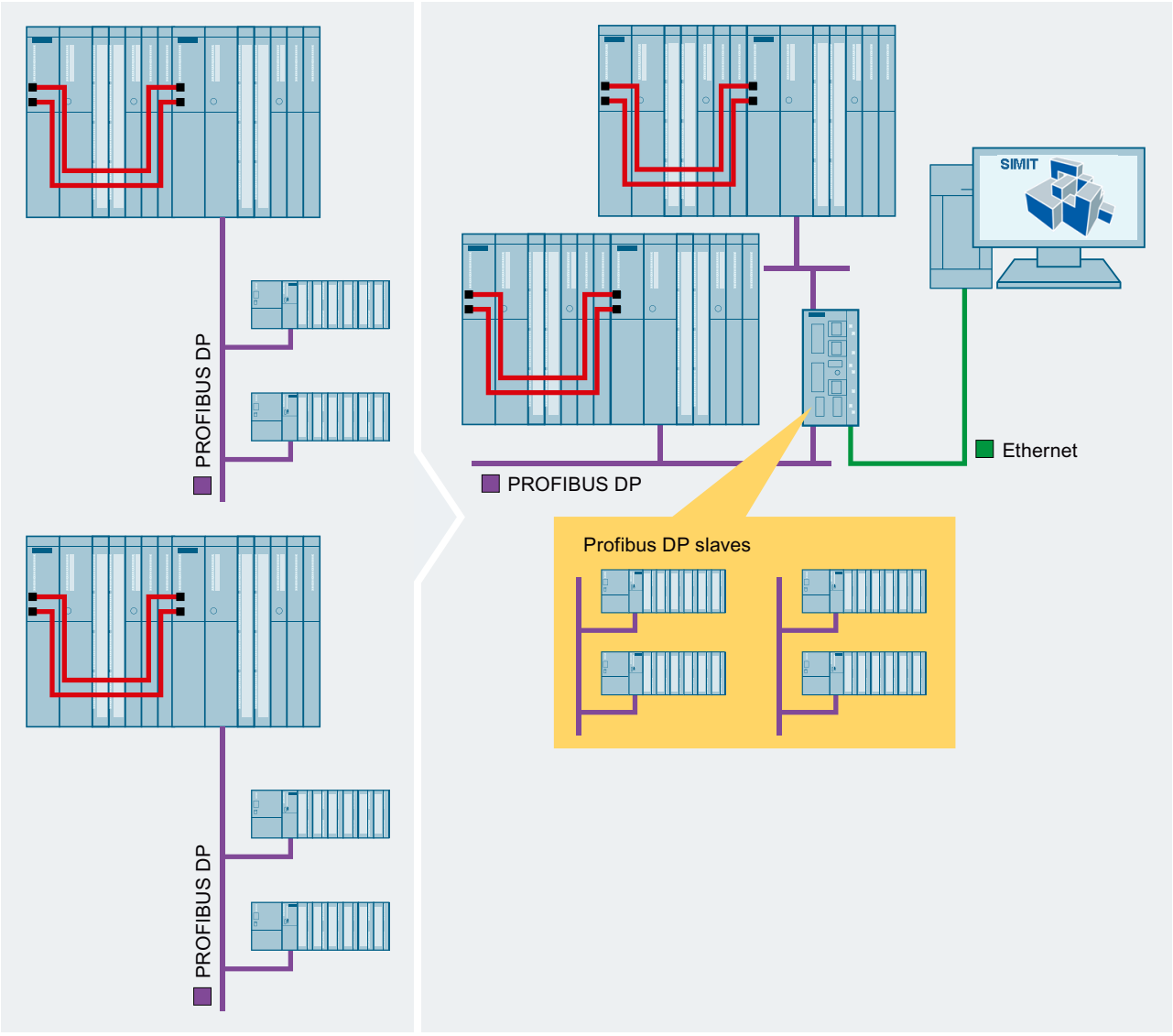
The redundant controller 1 is connected with the left PROFIBUS line 1 to port PB 1 of the SIMIT Unit. Port PB 2 of SIMIT Unit cannot be connected. PROFIBUS line 1 of the redundant controller 2 must not be connected to this SIMIT Unit. To connect PROFIBUS line 1 of the redundant controller 2, use a second SIMIT Unit.

The second CPU of controller 1 and the second CPU of controller 2 are not physically present, but are configured in STEP 7.


PROFIBUS DP
PROFIBUS DP
PROFIBUS DP slaves
SIMIT
Ethernet
PROFIBUS DP
PROFIBUS DP
PROFIBUS DP slaves

Two redundant controllers, each with a PROFIBUS DP line configured at one end only

The redundant controller 1 is connected with the associated PROFIBUS line to port PB 1 of the SIMIT Unit. The redundant controller 2 is connected with the associated PROFIBUS line to port PB 2 of the SIMIT Unit.



Note

Note that the station must actually be configured at one end only in STEP 7.

See also

Redundant configurations (H systems) (Page 89)

2.5.3 Supported PROFINET IO configurations

Introduction

Typical PROFINET IO configurations that are supported by SIMIT are listed below. In each figure below, you can see the configuration of the automation system on the left and the layout with SIMIT on the right.

- A controller with a PROFINET IO line
- A redundant controller with a redundant PROFINET IO line
- One controller with Shared Device
- Ring topology
- Topology on a CPU with a 2-port switch

Note

The IP address of the SIMIT Unit and the IP addresses of the simulated PROFINET IO devices must be located in different subnets.

Example:

- IP address of the SIMIT Unit: 192.16**9**.xxx.yyy, subnet mask: 255.255.0.0
- IP address of a simulated PROFINET IO device: 192.16**8**.xxx.yyy, subnet mask: 255.255.0.0

At least one digit of the first two segments of the IP address must be different.

Failure to comply with the rules for assigning the IP address can have the following consequences:

- Various simulated devices can be affected in such a way that they do not start up correctly.
- Faults can occur in components that are connected via the network to which the CTRL port is also connected, because the simulated devices are accessible via this CTRL port.

PROFINET components are simulated with the SIMIT UNIT PN box at the real PROFINET IO controller.

The following features are supported by SIMIT UNIT:

Feature	SIMIT UNIT PN 128	SIMIT UNIT PN 256
Order No.	9AE4120-2AA00	9AE4120-2AB00
Max. number of devices	128	256
PROFINET interfaces	1	2 z)
I/O data	✓	✓
Minimum cycle time	250 µs	250 µs
Data records	✓	✓
Device and module failure	✓	✓
Bus failure	✓	✓
Channel diagnostics	✓	✓
IRT class 2	✓	✓
IRT class 3	✓	✓

Feature	SIMIT UNIT PN 128	SIMIT UNIT PN 256
PROFIsafe V1.0 (PN/PN coupler)	✓	✓
PROFIsafe V2.4	✓	✓
PROFIsafe V2.6	✓	✓
Fast Startup	✓	✓
Advanced startup (CPU 1200/1500/410)	✓	✓
Multidevices (IE/PB Link)	restricted [2)]	restricted [2)]
I&M data	✓	✓
Shared Device	✓ [1)]	✓ [1)]
Redundancy MRP	✓ [3)]	✓ [3)]
System redundancy S2		✓
System redundancy R1		✓
System redundancy R2		
Save project permanently	✓	✓
Save project to USB/SD	✓	✓
Integrated switch		✓
Extended device diagnostics via SNMP		
Dynamic Frame Packaging		
Substitute values at CPU Stop	✓	✓
High-precision time stamping		
Shared I/O (MSI/MSO)		
Time Based I/O		
Fragmentation		
IRT with highest performance		
Isochronous mode		
Switched S1		✓
MRPD		
I-device	✓	✓

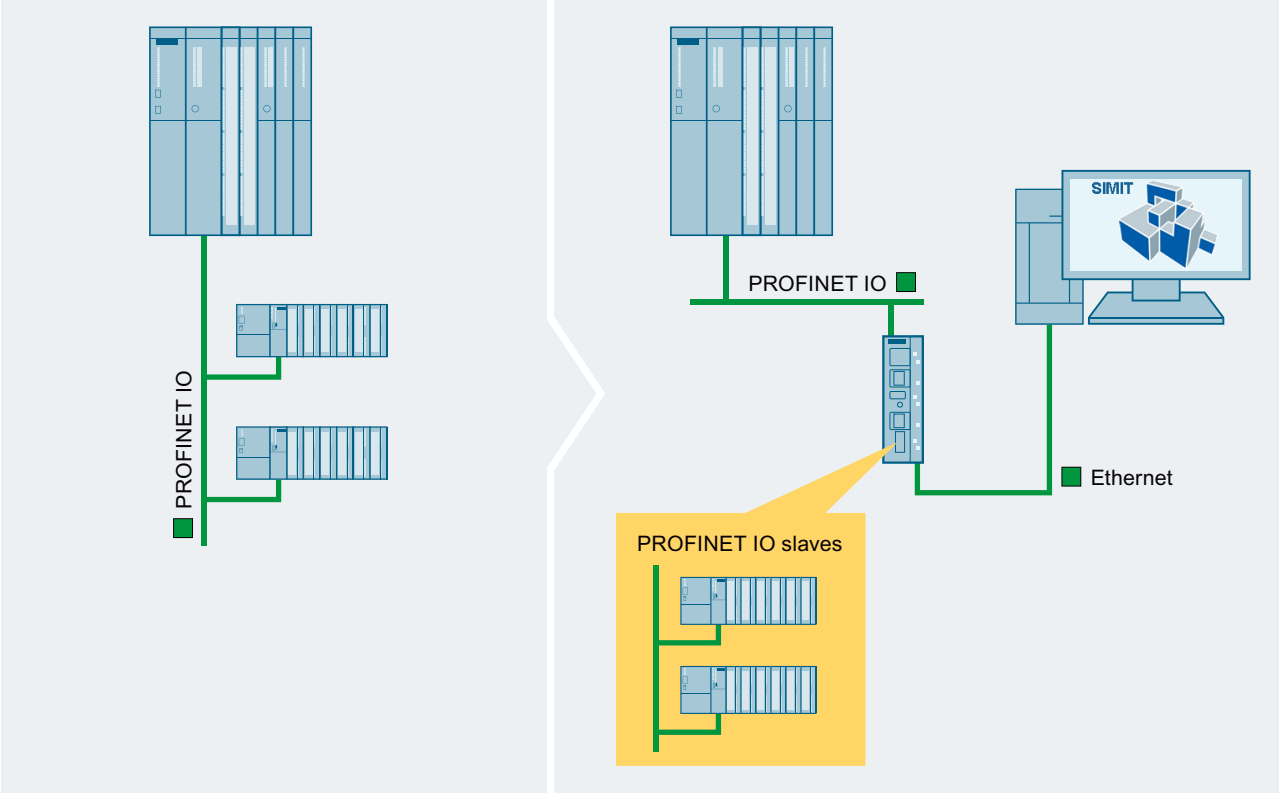
1) Projects with Shared Devices can be downloaded to SIMIT UNIT PN, but only access from a PNIO controller is supported.

2) For some devices, besides PNIO communication, configuration and diagnostic data are transferred from STEP 7 via the ISO protocol (e.g. IE/PB Link). The ISO protocol and the related simulation of the configuration and diagnostic data transfer of the devices involved is not supported by SIMIT UNIT PN. However, simulation of the PNIO portion of the communication still functions. Simulation of IE/PB Link on an S7-300 CPU does not function.

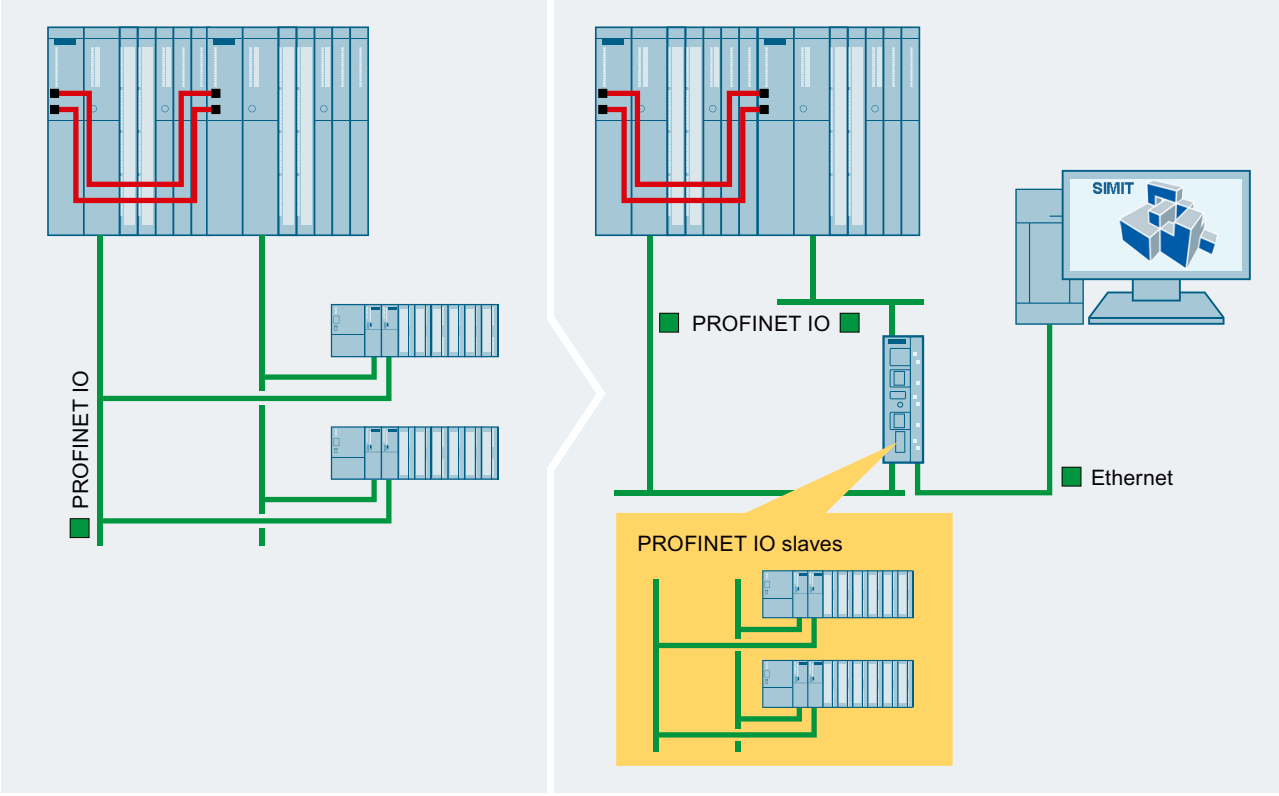
3) MRP client configurations are supported. MRP Manager configurations for the simulated device are not supported. With SIMIT UNIT PN 128, the ring cannot be closed because the second port is inactive.

z) Simulation of two separate PROFINET networks is not possible. Only simulation of a network with two interfaces to two PNIO controllers (e.g. for redundancy) is possible.

A controller with a PROFINET IO line



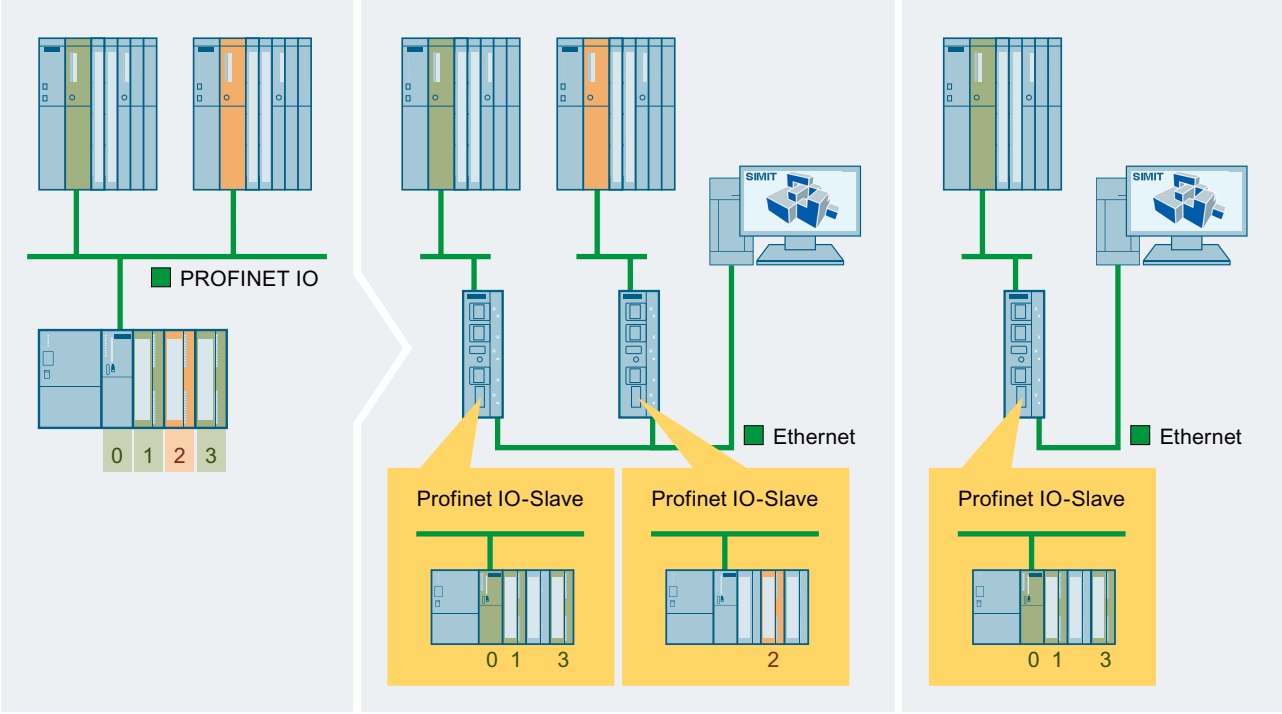
A redundant controller with a redundant PROFINET IO line



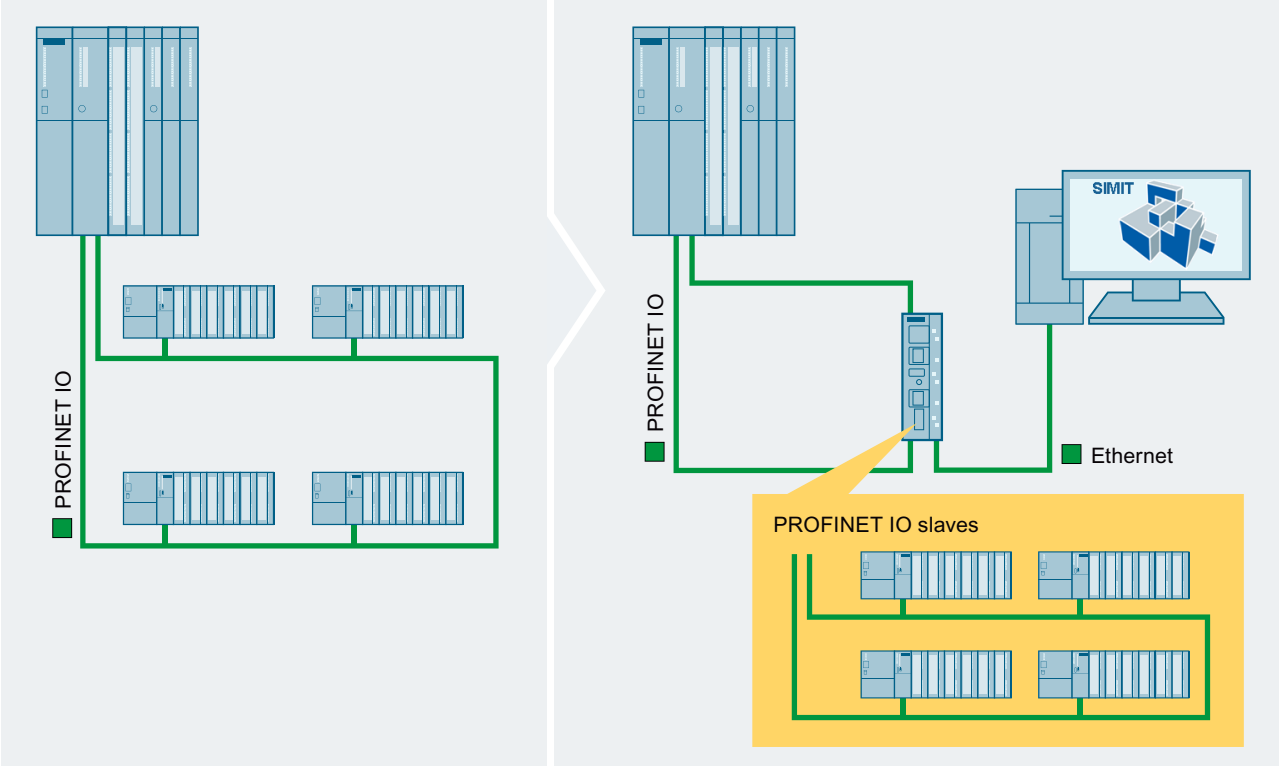
Note

For the S7-1500R/H-system, please note that the topology needs to be configured due to the additional MRP ring.

Controllers with Shared Device



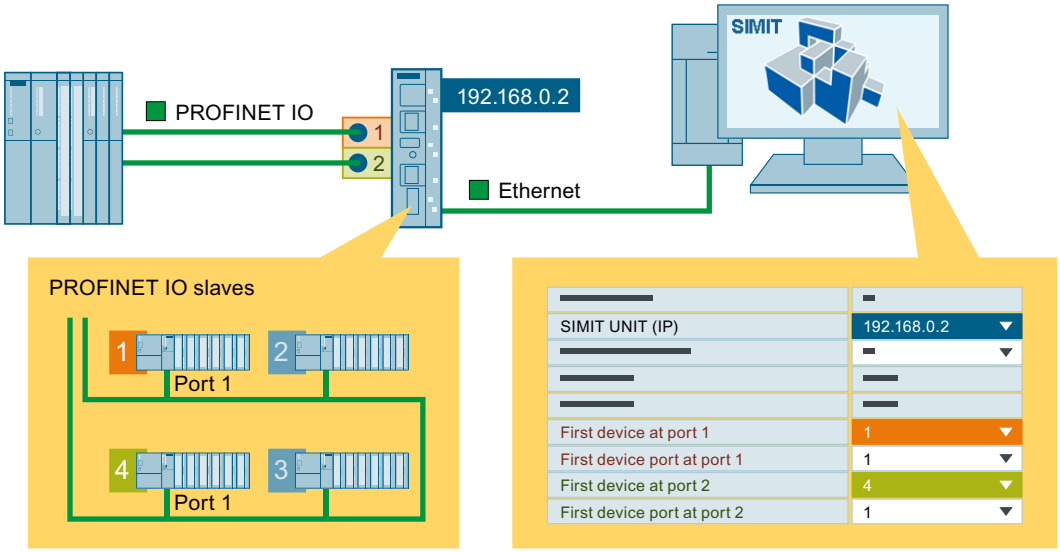
Ring topology



Note

For ring configurations, SIMIT UNIT PN256 only supports the role "MRP Client". It is not possible to simulate a device in the "MRP Manager" role. Therefore, it must be guaranteed that a real MRP manager is present in the test setup. This could be the CPU or a physically present switch, for example. If there is only one MRP manager in the project that cannot be physically present, STEP 7 or the TIA project must be adapted so that the CPU takes on the "MRP Manager" role. Alternatively, the ring should not be closed.

The figure below is a schematic representation of the necessary configuration settings in SIMIT:



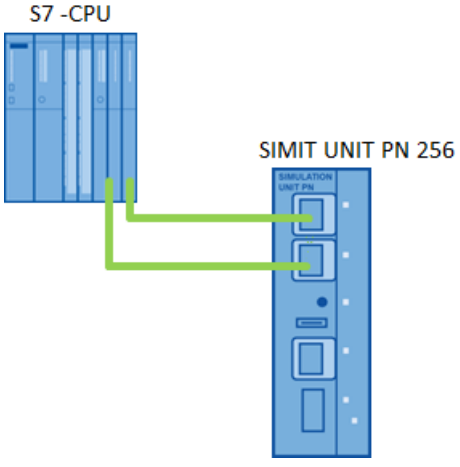
Topology on a CPU with a 2-port switch

A network topology is configured in the hardware configuration in which PROFINET devices are connected to two ports of a CPU that belong to the same switch (e.g. ports 1 and 2 of the X8 interface of a CPU410).

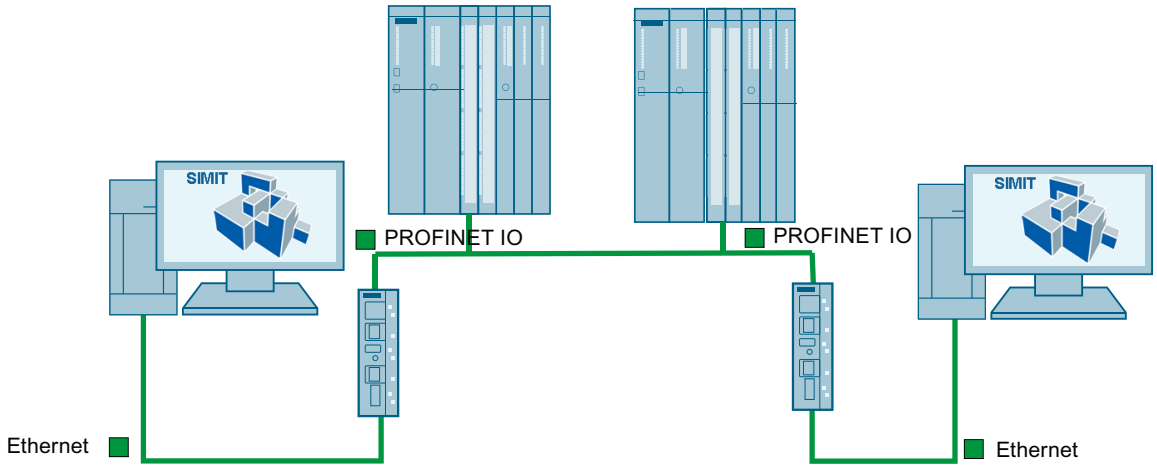
Both ports of the CPU are connected to the two interfaces P1 and P2 of a SIMIT UNIT PN 256.

Please note that only one PROFINET system per SIMIT UNIT PN is possible. If there are multiple PROFINET systems, the corresponding number of SIMIT UNIT PNs need to be used.

The SIMIT UNIT PN 128 has only one active port and therefore cannot be used for this configuration.



2 plants controlled with 2 SIMIT UNITS and connected with PN networks



Note

If every SIMIT UNIT PN is controlled from a separate PC, the simulated devices need to receive different MAC addresses. For this purpose, the MAC address of a PC in the file: "C:\Users \<username>\AppData\Roaming\SIMULATIONUnit\CFG\MACAdresses.xml" needs to be adapted. Afterwards, a re-import is required.

See also

Properties of the SIMIT Unit coupling (Page 113)

Redundant and fail-safe systems (Page 89)

Slave/I-device (Page 88)

2.5.4 Slave/I-device

Simulation of I-slaves/I-devices

SIMIT supports the simulation of I-slave/I-device using SIMIT Unit PB / PN.

The slave/device functionality with data exchange is supported. The I-slave/I-device is handled and imported like a normal slave/device.

As of SIMIT V10.2, an I-slave/I-device is recognized automatically during import. No GSD or GSDML file is created (or imported into SIMIT) in the process.

This also applies to fail-safe I-devices.

Exception: Fail-safe SIMOTION I-device. A GSDML is required for this. Refer to the STEP 7 or TIA documentation for instructions on creating the GSDML.

2.5.5 Redundant and fail-safe systems

2.5.5.1 Redundant configurations (H systems)

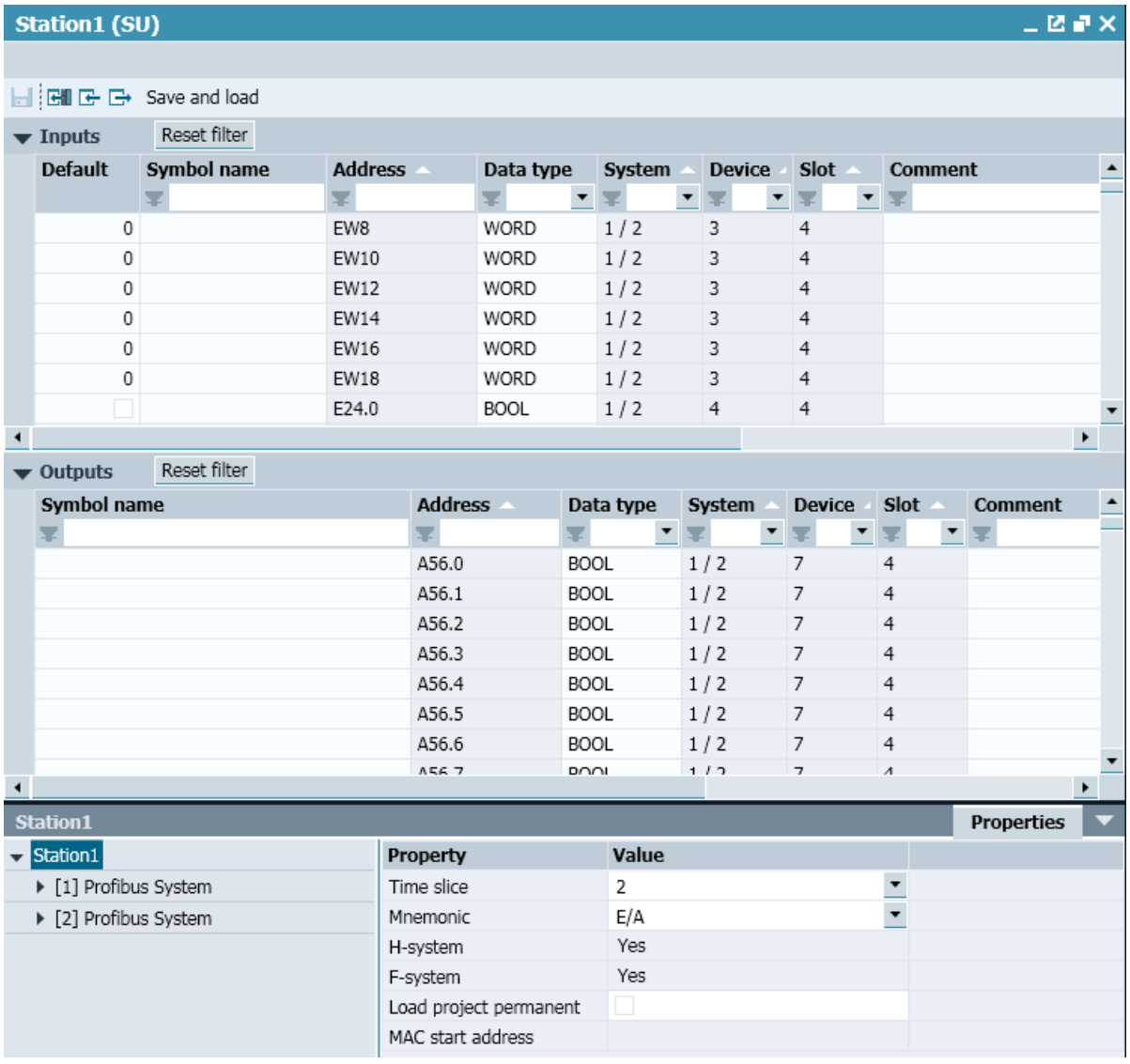
Introduction

SIMIT supports the simulation of redundant systems (H-systems). In a connected redundant system, all redundant diagnostic signals required are generated by the SIMIT Unit. This means you can also switch your controller from master to standby and back. You can see whether or not a slave is redundant under "H-system" in the slave property view.

DP/DP couplers can also be included as a Y-link in redundant systems. The coupling signals of bus nodes that are connected behind the Y-link are also available in SIMIT in the same way as the coupling signals of nodes that are connected directly to the redundant bus.

Redundant PROFIBUS system

Only the signals of the first redundant system (channel 0) are used as coupling signals in SIMIT, irrespective of which of the two buses is active. Bus switchovers are handled by SIMIT: The assignment of the coupling signals to the active channel is performed automatically. The redundant signals from the second channel are not listed. However, the signal list indicates that these signals are assigned to two master systems (1 and 2).



Redundant PROFINET system

Note

The simulation of a redundant PROFINET system is only supported with SIMIT Unit PN 256.

Only one line of the redundant system is created in SIMIT.

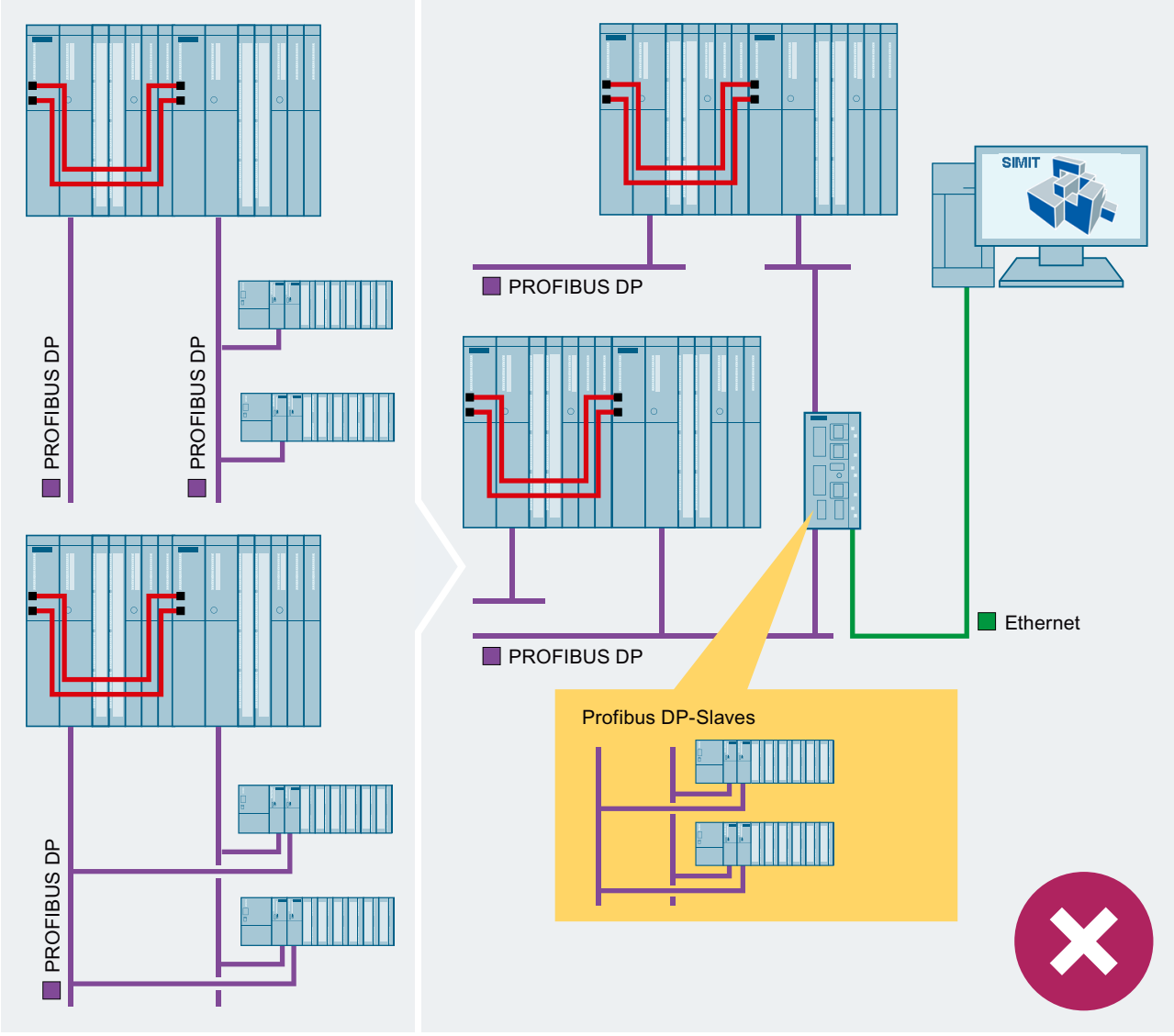
When you simulate R1 redundancy, disconnect the ports in the simulation project.

For the S7-1500R/H-system, the maximum number of simulated devices is limited to 128.

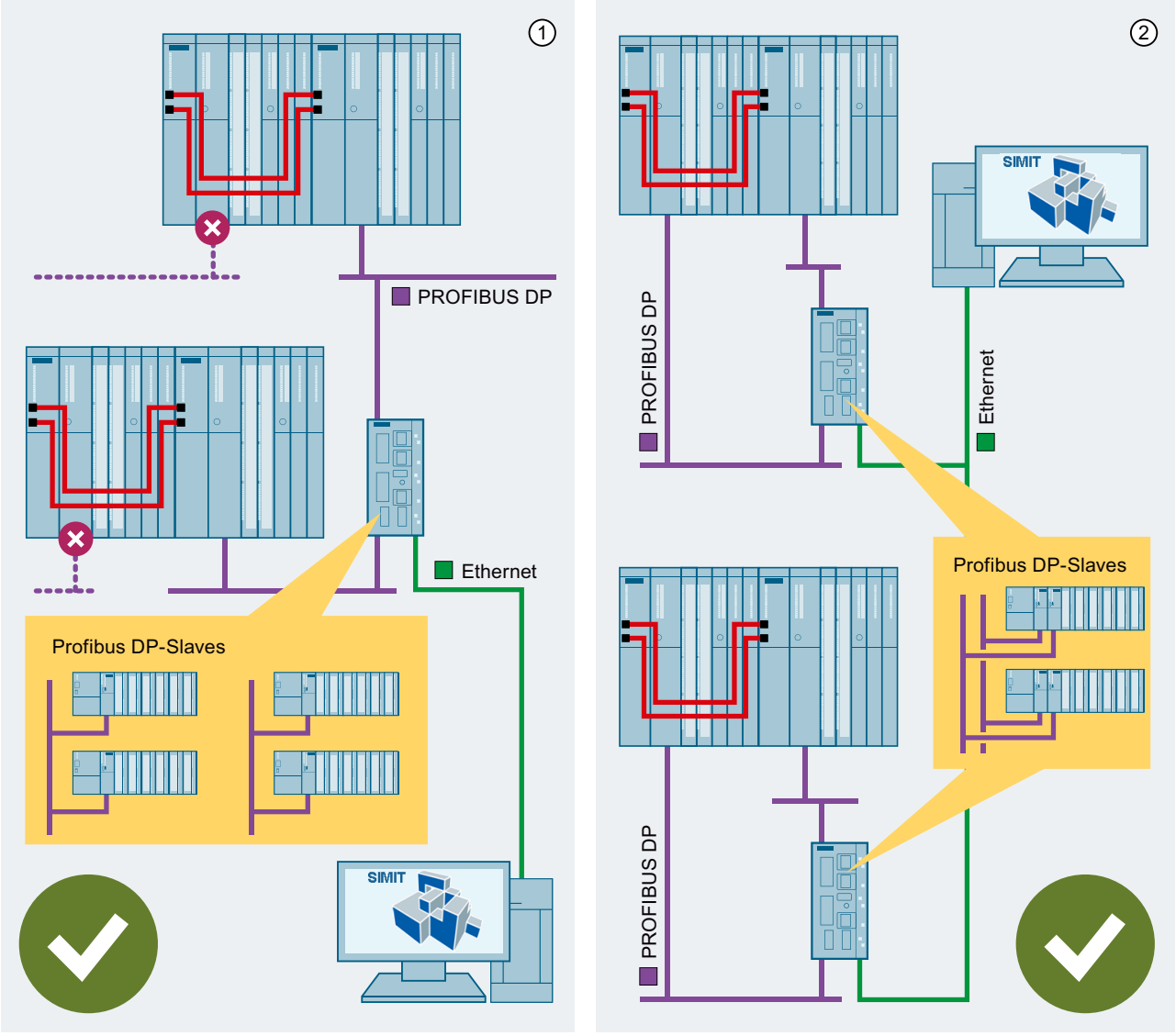
However, SIMIT UNIT PN 256 is required for this.

Setup example

When you configure a coupling to a redundant system, one SIMIT Unit is required for each redundant system. Two redundant systems are configured in the example below. Two lines are configured at each redundant system, but the distributed I/O is only connected to one line each. The following configuration is not permitted with only one SIMIT unit:



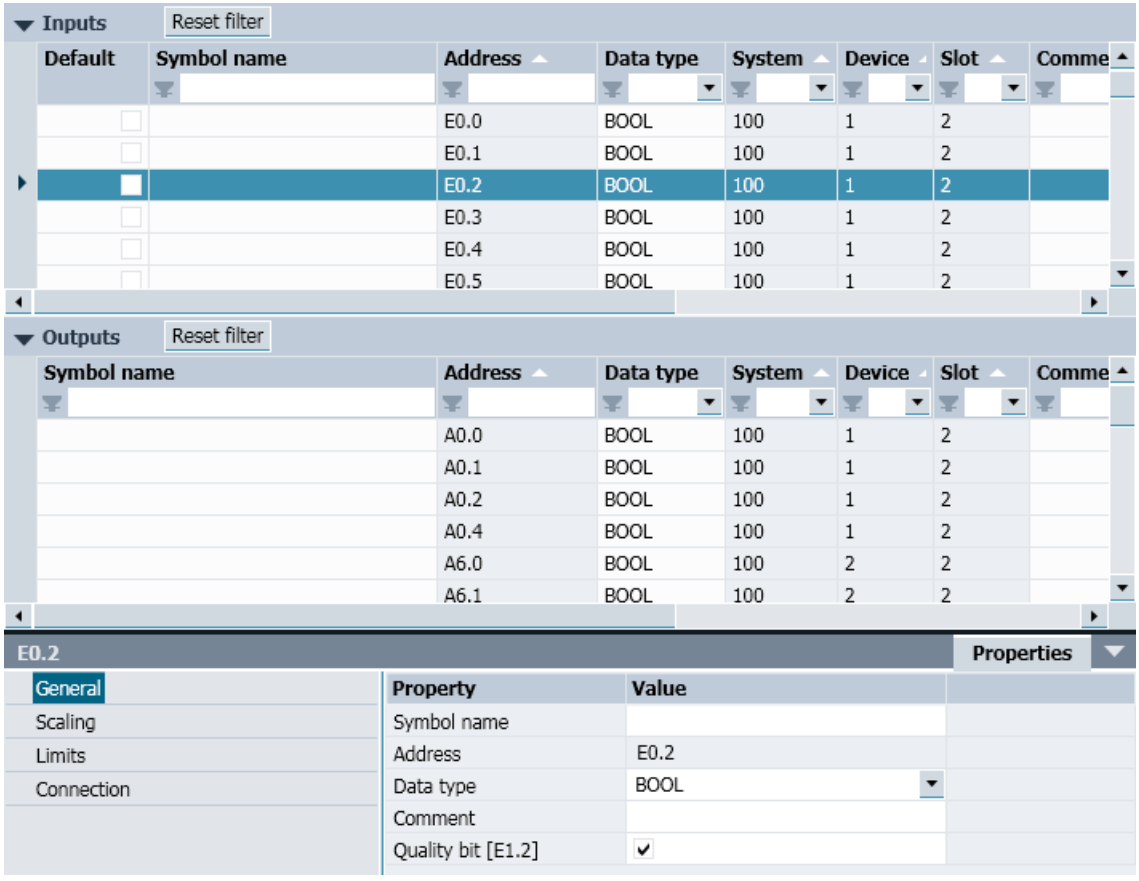
The figure below shows the two possible solutions:



① You remove the unused line in the hardware configuration.
② You use an additional SIMIT Unit.

2.5.5.2 Fail-safe configurations (F systems)

When you import a fail-safe configuration, the corresponding quality signals for all binary fail-safe signals are automatically generated and displayed in the property view of the fail-safe signal.



All quality signals set to "1" are valid by default. To set a signal to "invalid", set the corresponding quality signal to "0" and deselect the check box under "Quality bit" in the property view. This can be used, for example, to test the reaction of your control program to signal interference.

Note

After loading the interface module with a fail-safe configuration, the connected SIMATIC CPU needs to be restarted.

2.5.5.3 Redundant and fail-safe configurations (H/F systems)

Redundant and fail-safe systems (HF-systems) can also be simulated. The signals and quality signals are processed and displayed as described in the following sections:

- Redundant configurations (H systems) (Page 89)
- Fail-safe configurations (F systems) (Page 93)

The following mixed configurations can also be simulated:

- Fail-safe and non-fail-safe bus nodes
- Redundant and non-redundant connected bus nodes

2.5.6 Setting up and managing SIMIT Unit

2.5.6.1 Configuring a SIMIT Unit

Requirement

- SIMIT Unit can be accessed in the network.
- SIMIT is open.

Procedure

Follow these steps to configure a SIMIT Unit:

1. Select the "SU administration" command from the "Options" menu.
   The "SU administration" dialog box opens. The system automatically searches for available SIMIT units in the network.
2. Enter the following information for the SIMIT Unit:
   - Name
   - IP address
   - Subnet mask

Result

The SIMIT Unit is configured. The configuration data are saved in the SIMIT Unit.

See also

Operating principle of the SIMIT Unit coupling (Page 65)

"SU administration" dialog box (Page 993)

Creating a coupling of the "SIMIT Unit" type (Page 97)

2.5.6.2 Importing device description file to SIMIT

Introduction

If data for a device is already available in the delivered SIMIT software, it is shown in the "SU administration" as "System". If you import the GSD or GSDML files to a device, they are displayed in the "SU administration" as "User". The "User" information generally takes precedence over the "System" information.

Requirement

- SIMIT is open.
- Device description file is available.

Procedure

Follow these steps to import a device description file:

1. Select the "SU administration" command from the "Options" menu.
   The "SU administration" dialog box opens. The system automatically searches for available SIMIT units in the network.
2. Select the tab for importing a device description file.
3. Click on "Import" to open the device description data.

Result

The additional device information is imported and listed. The device information is saved on the PC on which SIMIT is installed.

See also

"SU administration" dialog box (Page 993)

Configuring a SIMIT Unit (Page 94)

2.5.6.3 Updating the firmware of the SIMIT Unit

Requirement

- SIMIT Unit is configured.
- SIMIT is open.
- Firmware file is available.

Procedure

NOTICE
Damage of SIMIT Unit possible during firmware update Observe the following rules for the duration of the firmware update: • Do not switch off the SIMIT Unit. • Do not restart the SIMIT Unit. • Do not remove or connect any cables from or to the SIMIT Unit. To ensure an uninterruptible power supply, connect the SIMIT Unit to a UPS.

Follow these steps to update the firmware of the SIMIT Unit:

1. Select the "SU administration" command from the "Options" menu.
   The "SU administration" dialog box opens. The system automatically searches for available SIMIT units in the network.
2. Select the desired SIMIT Unit.
3. Click "Firmware update".
4. Select the firmware file.

Result

The firmware of the SIMIT Unit is updated. Once the firmware update is complete, update the overview of SIMIT Units under "SU administration".

See also

Configuring a SIMIT Unit (Page 94)

"SU administration" dialog box (Page 993)

2.5.6.4 Resetting SIMIT Unit

Requirement

- SIMIT is open.
- A SIMIT Unit is accessible in the network.

Procedure

To reset a SIMIT Unit, proceed as follows:

1. Select the "SU administration" command from the "Options" menu.
   The "SU administration" dialog box opens.
   A search for reachable SIMIT Units is automatically started in the network.
2. Select the desired SIMIT Unit.
3. Select "Reset" in the shortcut menu and confirm the security prompt with "Yes".

Result

The SIMIT Unit has been reset.

This procedure corresponds to pressing the reset switch on the SIMIT Unit.

2.5.7 Configuring a coupling of the "SIMIT Unit" type

2.5.7.1 Creating a coupling of the "SIMIT Unit" type

Introduction

You can create a coupling of the "SIMIT Unit" type in a SIMIT project. You create "stations" under the SIMIT Unit to which the hardware configuration of a station is imported from STEP 7.

A maximum of 32 stations can be created.

Requirement

- Project is open in SIMIT.

Procedure

Follow these steps to create a coupling of the "SIMIT Unit" type:

1. Add a new coupling in the project tree under "Couplings".
   The "Selection" dialog box opens.
2. Select the coupling type "SIMIT Unit".

Result

The "SIMIT Unit" coupling is created together with a station in the project tree.

See also

Configuring a SIMIT Unit (Page 94)

2.5.7.2 Creating a station

Requirement

- Project is open.
- Coupling of the "SIMIT Unit" type has been created.
- Hardware configuration from STEP 7 is available.

Procedure

Follow these steps to create a station:

1. Add a new station under "SIMIT Unit" in the project tree.
2. Open the station.
   The "SU import" dialog box will open.
3. Configure the import of the stations from the hardware configuration.

Result

You can find the result after import of the hardware configuration in Importing hardware configuration to station (Page 99).

See also

"SU Import" dialog box (Page 983)

2.5.7.3 Copying system data blocks

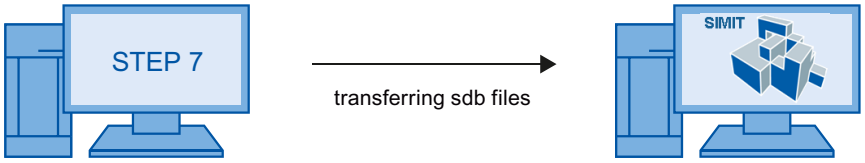
Introduction

If STEP 7 and SIMIT are installed on different PCs, transfer the system data blocks with a software package provided by SIMIT:

- UnlockHWConfig.exe
- CopyHWConfig.exe

Neither program requires installation and both can be accessed from anywhere. The programs are saved in the following folders:

- PC with installed SIMIT: Windows start menu under "SIMIT > Tools"



UnlockHWConfig.exe

In STEP 7, the sdb files are by default deleted following compilation of the hardware configuration. To prevent this permanently, launch the program "UnlockHWConfig.exe" once on the STEP 7 PC. Note that you must run this program with administrator rights.

CopyHWConfig.exe

To copy the system data blocks, use the program "CopyHWConfig.exe". By default, the following folders are set as source folders:

- STEP 7 V5.5 and later: Temporary STEP 7 storage folder
- STEP 7 V16 and higher: "C:\Users\Public\Documents\TIAExport\HwConfiguration\DOWN\"

2.5.7.4 Importing hardware configuration to station

Note

Configuration in Run (CiR)

"Configuration in Run" is not supported by SIMIT. Import the system data blocks again if you have changed the hardware configuration in STEP 7.

Introduction

You import the hardware configuration directly from STEP 7/PCSneo or from the copied system data blocks. System data blocks are created each time you compile the hardware configuration in STEP 7/PCSneo. Import the hardware configuration to the station in SIMIT again if you have changed the hardware configuration in STEP 7/PCSneo.

STEP 7 version	SIMATIC controllers supported	Format of the system data blocks
V5.5 and higher[1]	• S7-300 / S7-400	• SDB
as of V13 (TIA Portal)[2]	• S7-300 / S7-400 • S7-1200 / S7-1500	• SDB • OMS
PCSneo[3]	• S7-410	• SDB

1 Standard storage folder for system data blocks: "<STEP 7 installation folder>\S7Tmp\SDBDATA \S7hwcnfx\DOWN\r00s0x"

2 Standard storage folder for system data blocks: "C:\Users\Public\Documents\TIAExport \HwConfiguration\DOWN\"

3 Standard storage folder for system data blocks: See PCSneo documentation

Exporting the hardware configuration from STEP 7 V5.5 and higher

Export the hardware configuration from STEP 7 in the following formats:

- SDB
- CFG

Importing the hardware configuration of third-party systems

SIMIT supports the import of the hardware configuration of third-party systems in *.XML format. You can find additional information under "XML import interface for hardware configuration of third-party systems (Page 116)".

Requirement

- Coupling of the "SIMIT Unit" type has been created.
- Station has been created under the coupling.
- Station hardware configuration from STEP 7 is available.
- Hardware configuration in STEP 7 has been compiled and loaded.
- Station is open in the coupling editor.

Procedure

Follow these steps to import the hardware configuration to a station:

1. Open the dialog box for importing the hardware configuration in the Coupling Editor.
2. Select the source from which you want to import the hardware configuration.
3. If you import the hardware configuration from STEP 7 V5.5 and higher, also select the configuration file in *.CFG format.
4. Select the devices.
5. Click "Import".

Result

The hardware configuration of the station is imported.

See also

Editing the signals of a coupling (Page 210)

Creating a station (Page 98)

Configuring the station (Page 107)

"SU Import" dialog box (Page 983)

2.5.7.5 Shared devices

Introduction

SIMIT supports simulation of shared devices.

Requirement

- Coupling of the type "SIMIT Unit" has been created.
- Station has been created under the coupling.
- Hardware configuration of the stations from STEP 7 exists.
- Station with the header data for the Shared Device is known.
- Hardware configuration is compiled and loaded in STEP 7.
- Station is open in the coupling editor.

Procedure

To import the hardware configuration for shared devices into a station, proceed as follows:

1. Import the hardware configuration of the CPU into the station without header data:
   - In the coupling editor, open the dialog box for importing the hardware configuration.
   - Select the source.
   - If you are importing the hardware configuration of an S7-300 or S7-400 based on STEP 7 or PCS 7, also specify the configuration file.
   - Select the devices.
   - Click "Import".

   The "Incomplete import of shared devices detected" message is displayed.
2. Import the hardware configuration of the CPU which contains the header data of the Shared Device into the station:
   - In the coupling editor, open the dialog box for importing the hardware configuration.
   - Select the source.
   - If you are importing the hardware configuration of an S7-300 or S7-400 based on STEP 7 or PCS 7, also specify the configuration file.
   - Activate the "Additional information for Shared Devices" option.
   - Select the devices.
   - Click "Import".

Result

The hardware configuration of the station from STEP 7 is imported.

2.5.7.6 Modules that cannot be uniquely identified

If different modules are available for a module type, the assignment of the modules is adapted in SIMIT.

You change the assignment of modules in the "Properties" window of the modules.

Properties of the SIMIT Unit coupling (Page 113)

List of module types

Below is a list of module types for which multiple modules are available for selection.

Table 2-1 PROFIBUS

Type (slave family)	Possible selections in SIMIT	MLFB	Ambiguous for S7 V5.5 (SDB +CFG)	Ambiguous TIA / PCSneo (SDB / OMS)
ET 200S	6ES7 131-4BB0x-0Ax0 2DI DC24V	6ES7 131-4BB00-0AB0	No	Yes
		6ES7 131-4BB01-0AB0		
		6ES7 131-4BB00-0AA0		
		6ES7 131-4BB01-0AA0		
	6ES7 131-4xB00-0AB0 2DI 120/230V	6ES7 131-4EB00-0AB0		
		6ES7 131-4FB00-0AB0		
	6ES7 131-4BD0x-0Ax0 4DI DC24V	6ES7 131-4BD00-0AA0		
		6ES7 131-4BD01-0AA0		
		6ES7 131-4BD00-0AB0		
		6ES7 131-4BD01-0AB0		
		6ES7 131-4BD50-0AA0		
		6ES7 131-4BD51-0AA0		
	6ES7 131-4BF00-0AA0 8DI	6ES7 131-4BF00-0AA0		
		6ES7 131-4BF50-0AA0		
	8DI	*Display only		
ET 200S	2 words IN	*Display only	No	Yes
	SSI Module Fast	6ES7 138-4DB00-0AB0 Fast		
		6ES7 138-4DB01-0AB0 Fast		
		6ES7 138-4DB02-0AB0 Fast		
		6ES7 138-4DB03-0AB0 Fast		
DP/DP Coupler (6ES7 158-0AD00-0XA0) (6ES7 158-0AD01-0XA0)	Sim F_SENDDP	Universal module 12/6 bytes	Yes	Yes
	12 bytes IN 6 bytes OUT	Universal module 12/6 bytes		
DP/DP Coupler (6ES7 158-0AD00-0XA0) (6ES7 158-0AD01-0XA0)	Sim F_RCVDP	Universal module 6/12 bytes	Yes	Yes
	6 bytes IN 12 bytes OUT	Universal module 6/12 bytes		

ET 200S HighFeature (Cu)	2 words IN / OUT	*Display only	No	Yes
	1 SI ASCII (4 bytes)	6ES7 138-4DF0*-0AB0		
	1 SI 3964(R) (4 bytes)	6ES7 138-4DF0*-0AB0		
ET 200S HighFeature (Cu)	4 words IN / OUT	*Display only	No	Yes
	1 SSI module	6ES7 138-4DB0*-0AB0		
	1 SI ASCII (8 bytes)	6ES7 138-4DF0*-0AB0		
	1 SI ASCII V1.3 (8 bytes)	6ES7 138-4DF0*-0AB0		
	1 SI 3964(R) (8 bytes)	6ES7 138-4DF0*-0AB0		
ET 200S HighFeature (Cu)	16 words IN 16 words OUT	*Display only	No	Yes
	1 SI 3964(R) (32 bytes)	6ES7 138-4DF0*-0AB0		
SIMOCODE pro V	Basic Type 2	3UF7 010-1A*00-0	Yes	Yes
	Basic Type 2 (words view)	*Display only		
ET 200pro (IM154-1)	F-switch	6ES7 148-4FS00-0AB0	No	Yes
	RF170C	6GT2002-0HD00		
ET 200pro (IM154-2AA01 HF)	F-switch	6ES7 148-4FS00-0AB0	No	Yes
	RF170C	6GT2002-0HD00		
ET 200pro (IM154-2AA01 HF)	4/8 F-DI / 4 F-DO DC24V/2A	6ES7 148-4FC00-0AB0	No	Yes
	F-switch	6ES7 148-4FS00-0AB0		
SM module (ET 200M)	6ES7 331-7TF0x-0AB0 AI8x16Bit HART/TC (AI/AO)	6ES7 331-7TF00-0AB0	Yes	Yes
		6ES7 331-7TF01-0AB0		
	6ES7 331-7TB00-0AB0 AI2xHART (CRC)	6ES7 331-7TB00-0AB0		
	6ES7 331-7TB00-0AB0 AI2xHART (FLOAT)	6ES7 331-7TB00-0AB0		
SM module (ET 200M)	AO2 HART 332-5TB00-0AB0 (CRC)	6ES7 332-5TB00-0AB0	Yes	Yes
	AO2 HART 332-5TB00-0AB0 (FLOAT)	6ES7 332-5TB00-0AB0		
	6ES7 332-5TB00-0AB0 AO2xHART (AI/AO)	6ES7 332-5TB00-0AB0		
SM module (ET 200M)	6ES7 350-2AH00-0AE0 FM 350-2; 16ByteI/O	6ES7 350-2AH00-0AE0	Yes	Yes
	6ES7 350-2AH00-0AE0 FM 350-2; 8x Counter	6ES7 350-2AH00-0AE0		
	6ES7 350-2AH00-0AE0 FM 350-2; 4x Counter	6ES7 350-2AH00-0AE0		
*Display only: Only the signal splitting is changed when these modules are adapted.				

Table 2-2 PROFINET

Type (device family)	Possible selections in SIMIT	MLFB	Ambiguous for S7 V5.5 (SDB +CFG)	Ambiguous for TIA (SDB/ OMS)
DP/DP Coupler (6ES7 158-3AD00-0XA0) (6ES7 158-3AD01-0XA0)	PROFIsafe IN/OUT 12 bytes/6 bytes	IN/OUT 12 bytes/6 bytes	Yes	Yes
	12 bytes Input/6 bytes Output (F_Receive)	IN/OUT 12 bytes/6 bytes		
DP/DP Coupler (6ES7 158-3AD00-0XA0) (6ES7 158-3AD01-0XA0)	PROFIsafe IN/OUT 6 bytes/12 bytes	IN/OUT 6 bytes/12 bytes	Yes	Yes
	6 bytes Input/12 bytes Output (F_Send)	IN/OUT 6 bytes/12 bytes		

2.5.7.7 DP/DP coupler

With the SU coupling, hardware configurations that contain fail-safe master-master communication (Profisafe V1) can also be used.

The counterpart of a real F_SENDDP / F_RCVDP block configured in the CPU is simulated with SIMIT and the SIMIT Unit .

The SIMIT Unit PROFIBUS or PROFINET simulates the configured DP/DP coupler or PN/PN coupler with the respective universal modules. The fail-safe communication is simulated over the user data of the module.

Import of the hardware in SIMIT is the same as in a standard project. The only thing that must be taken into consideration in the HW Config of the S7 project is that the universal modules in the DP/DP coupler or PN/PN coupler have the correct length for the fail-safe communication blocks.

```
+----------+ ----12 bytes----> +----------+
| F_SENDDP |                   | F_RECVDP |
+----------+ <----6 bytes----- +----------+
```

If an incorrect length is configured (e.g. a larger range), SIMIT recognizes the module as standard I/Q module.

A PROFIsafe_V1 function is automatically implemented in the SIMIT project.

Each F_SENDDP / F_RCVDP block in the real CPU has the matching counterpart in SIMIT.

To simulate the F_RCVDP, the correct DP_DP_ID only needs to be entered in the hardware view of the SU coupling for the respective module or submodule.

For the simulation of the F_SENDDP sender block, the DP_DP_ID also needs to be specified.

If the information is correct before the CPU start, the connection will be established successfully after starting the CPU.

If the CPU was already running before, the F_RCVDP block must be reintegrated into the CPU over the ACK_REI input.

NOTICE
If the connection is interrupted (for example by pulling the PROFIBUS cable), the CPU must be restarted for the reintegration. Otherwise, it does not reset its internal telegram counters.

The DP-DP-ID is a system-wide, unique connection ID. This parameter must be identical to the parameter of the same name of the partner block in the CPU.

Example

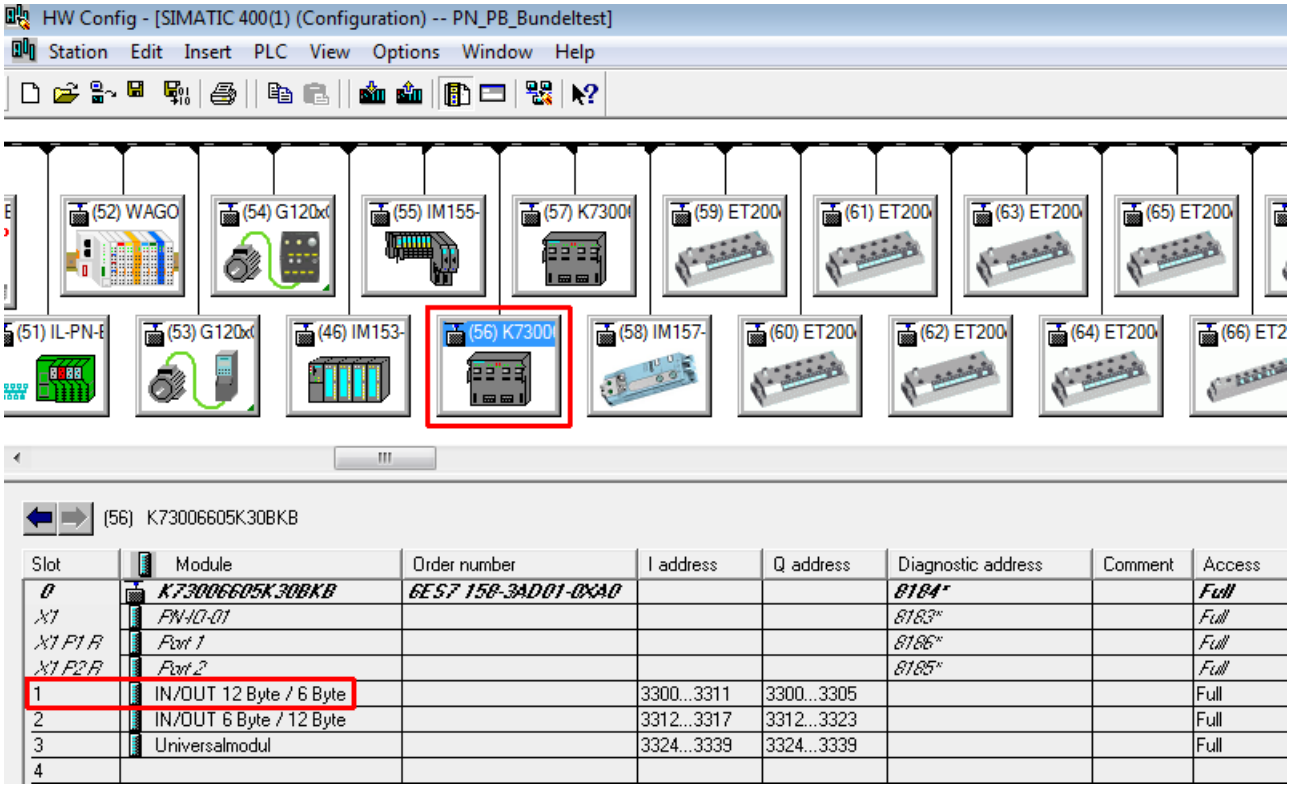


Figure 2-1 Hardware configuration: Slot 1 of the device 56 in the PROFINET is highlighted

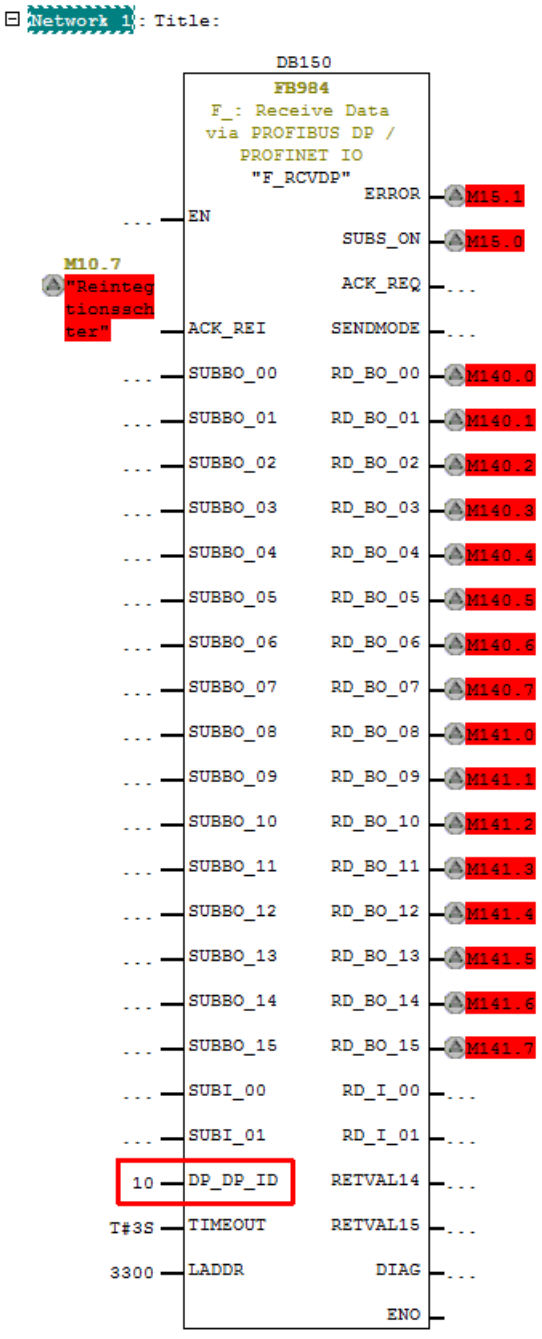
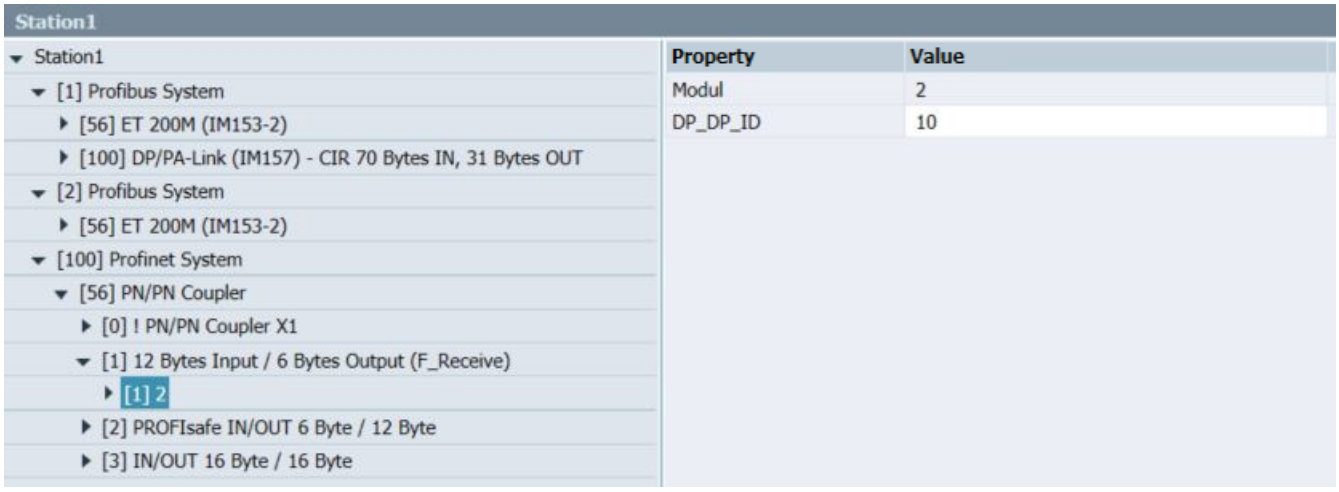


Figure 2-2 F_RCVDP: The DP-DP-ID has the value 10.

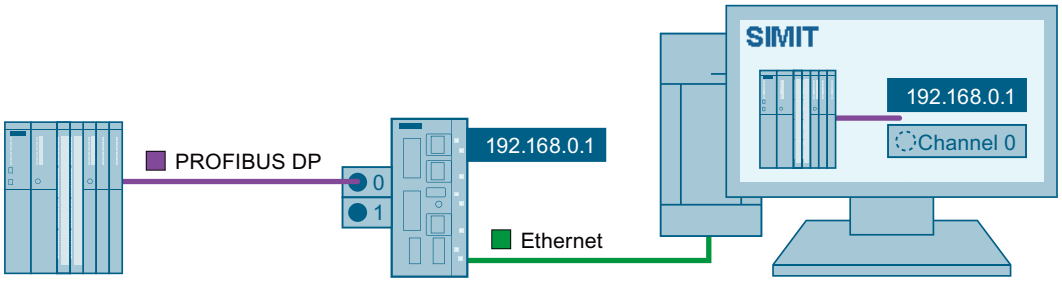
The same value, 10, is entered for the DP-DP-ID on the (sub)module as in the F_RCVDP.



2.5.7.8 Configuring the station

Introduction

The schematic diagram below shows what information is required for configuring a SIMIT Unit in SIMIT:



Requirement

- Station has been created.
- IP address of the SIMIT Unit is known.
- Hardware configuration has been imported.

Procedure

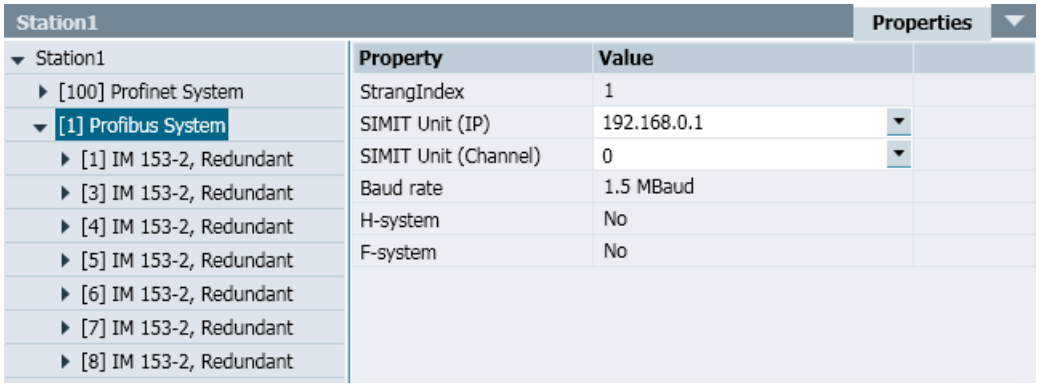
Follow these steps to configure a station:

1. Double-click on the required station in the project tree under "Couplings > SIMIT Unit".
2. In the properties window, select the "time slice" of the station.

3. In the properties window, assign each master system to a SIMIT unit under "Station":
   - Select the IP address of the SIMIT Unit.
   - Select the channel of the SIMIT Unit to which the master system is connected.
4. Save the configuration.

Result

The station is configured. The figure below shows the assignment of a PROFIBUS master system of a station to a SIMIT Unit:



See also

Loading configuration to station (Page 112)

Importing hardware configuration to station (Page 99)

Properties of the SIMIT Unit coupling (Page 113)

2.5.7.9 Simulation of alarms

Alarms

In SIMIT, you can simulate alarms for the following levels of a station:

- PROFIBUS
  Line, slave, module, channel
- PROFINET IO
  Line, device, module, submodule, channel

You simulate alarms either in the properties of a station or with the "SetInterrupt" component.

Note

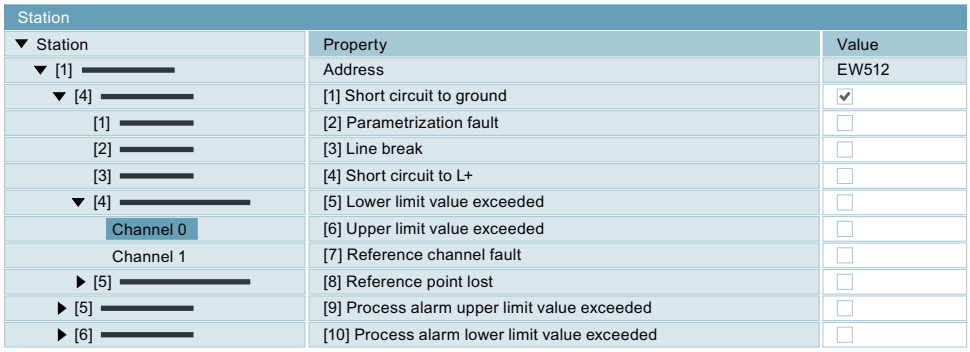
Response with migrated projects from SIMIT versions earlier than V10

To simulate alarms, import a current hardware configuration once.

Simulate alarm in the station properties

Different alarms can be simulated depending on the hardware configuration. The alarms that can be simulated for the selected component are displayed in the station properties.

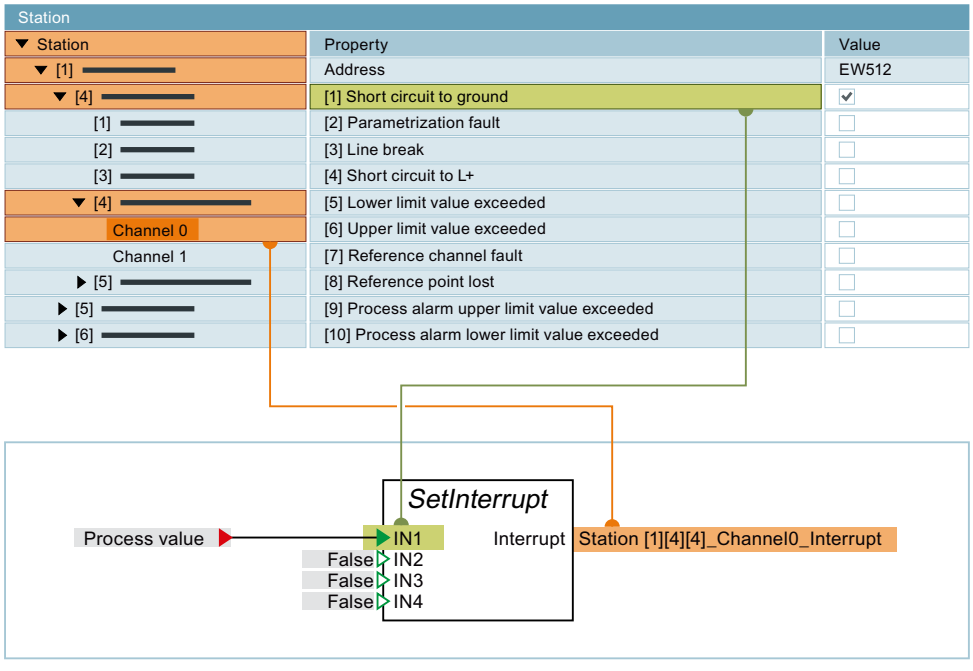
In the example below, the check box at "[1] Short-circuit to M" activates a short circuit to ground in Channel 0.



Simulate alarm with the component "SetInterrupt"

The number of the desired alarm in the properties determines the number of the input of the component "SetInterrupt".

In the example above, Channel 0 has the error "Short circuit to M" under number 1. You therefore connect the process value with the "IN1" input of the component "SetInterrupt". Channel 0 is part of the coupling Station, line 1, slave 4, module 4. You therefore connect the signal interrupt of Channel 0, that is "Station [1][4][4]_Channel0_Interrupt" to the output of the component "SetInterrupt". The signal is located in the "Signals" task card.

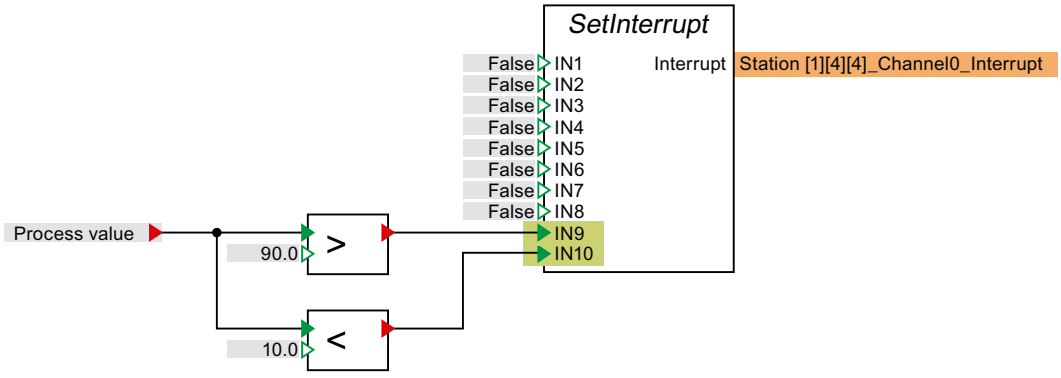


Hardware interrupts

SIMATIC PLCs use hardware interrupts to monitor binary or analog I/O signals: When an analog value is exceeded or undershot or when a binary value is set or reset, the cyclical operation of the controller is interrupted to execute the reaction of the user program to this interrupt. If and under which conditions a device or module triggers hardware interrupts is defined in the STEP 7 project.

Example: Simulation of a hardware interrupt for an analog value

The following figure shows a configuration example for the simulation of a hardware interrupt. The hardware interrupt is triggered when configured limits are exceeded or undershot. You use the component "Compare" to simulate the limit monitoring:



See also

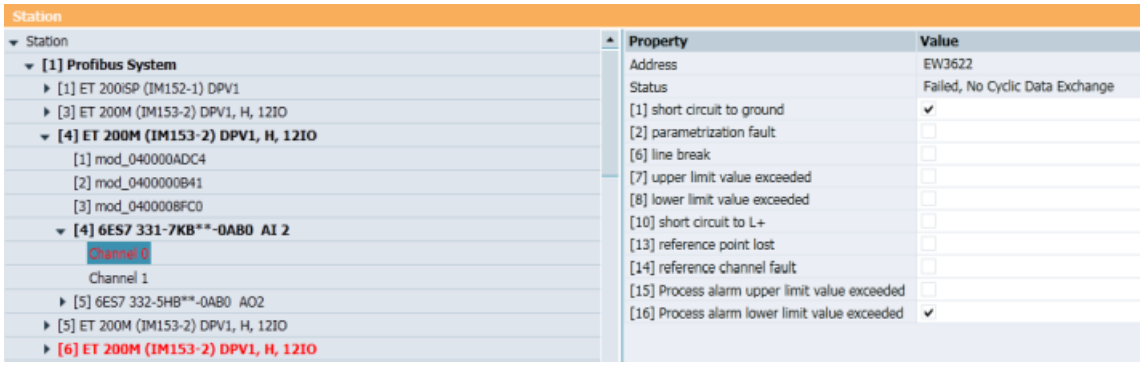
SetInterrupt (Page 579)

2.5.7.10 Device status

The device status is determined from within the SIMIT Unit hardware and shown in the hardware tree. The markings have the following meaning:

- Red: Error
- Blue: Warning
- Bold: Partial tree with errors or warnings

In the example below, an error exists in Channel 0. Channel 0 is therefore highlighted in red. The module [4], the slave [4] and the line [1] are shown in bold because a lower-level object has an error. There are also warnings for slave [6] and line [1]. This is why the slave and line are highlighted in blue. The status of the marked component is shown in the properties, e.g. "Failed, No Cyclic Data Exchange" or "Cyclic Data Exchange".



2.5.7.11 Disabling a line, device or module

Introduction

For example, to simulate the failure of a line, device or module during a simulation, disable the line, device or module in the coupling properties. In this way you can test, for example, the reaction of the controller to failure and return of a line, device or module.

Deactivation works while simulation is running and during configuration.

Requirement

- Coupling of the type "SIMIT Unit" has been created.
- Station is open in the coupling editor.
- Station is configured.

Procedure

To disable a line, a device or a module, proceed as follows:

1. To disable a line:
   - Select the line in the properties.
   - Select the option "Line failure".
2. To disable a device:
   - Select the device in the properties.
   - Select the option "Slave failure".
3. To disable a module:
   - Select the module in the properties.
   - Select the option "Pull module".

Result

The line, device or module has been disabled.

See also

Configuring the station (Page 107)

2.5.7.12 Loading configuration to station

Requirement

- Master system is connected to simulation.
- SIMIT Unit is connected to the PC on which SIMIT is installed.
- Station is configured.

Procedure

Note

Communication between the master system and controller is briefly interrupted during loading.

Follow these steps to load a configuration to a station:

1. Open the station.
2. If you want to also save the SIMIT project in the SIMIT Unit, select the "Load project permanently" option in the station properties.
3. Click "Save and load" in the coupling editor.

Result

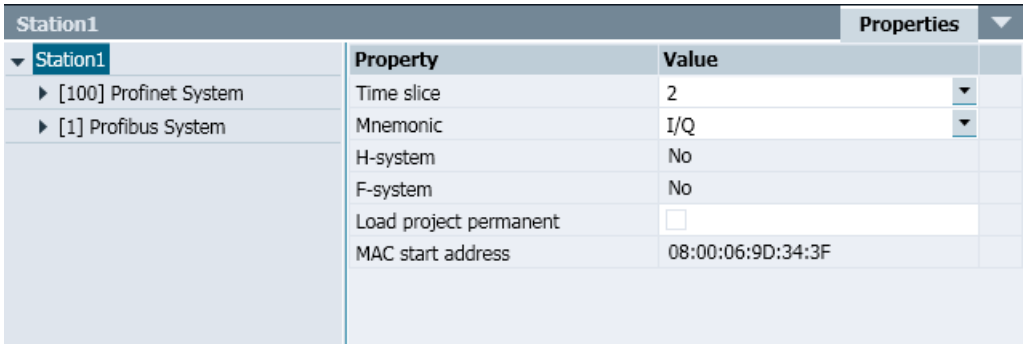
The configuration is loaded to the SIMIT Unit. You can launch simulation for this station once loading is complete.

See also

Configuring the station (Page 107)

2.5.7.13 Properties of the SIMIT Unit coupling

Properties of a station



- **Time slice**
  Here you set the cycle at which the coupling exchanges data. The assignment of absolute cycle times to the 8 available time slices is valid for the entire project. The default is time slice 2, which corresponds to a cycle of 100 ms.

  Note

  The time slice with the lowest cycle time always has the highest priority, regardless of the numbering.

- **Mnemonics**
  Specifies whether the German ("E/A") or English ("I/Q") term for the absolute addresses is used.

  Note

  Define the mnemonics immediately after import. If you subsequently change the mnemonics, signals already interconnected in a chart can no longer be assigned to the original address.

- **H system**
  Shows that the system is an H system.
- **F system**
  Shows that the system is an F system.
- **Load project permanent**
  Specifies that the simulation project is also saved in the SIMIT Unit. You can continue the simulation when the SIMIT Unit is restarted without having to load the simulation project to the SIMIT Unit once again.
  To delete the simulation project from the SIMIT Unit, disable the option and load the configuration to the SIMIT Unit.

General properties of a connection over PROFIBUS/PROFINET

- **LineIndex**
  Shows the number of the master system.
- **SIMIT UNIT (IP)**
  Specifies the IP address of the Simulation Unit to which the SIMIT PC is connected.
- **SIMIT Unit (channel)**
  Specifies the channel on which the controller is connected to the SIMIT Unit.
- **H system**
  Shows that the system is an H system.
- **F system**
  Shows that the system is an F system.

Specific properties of a connection over PROFINET

- **Split SU Ports**
  If this option is selected, the switch communication is suppressed during simulation of the topology on the SIMIT UNIT.
- **First device at port 1**
  Specifies the device that is configured as the first device in the network configuration as seen from the PROFINET controller.
- **First device port at port 1**
  Specifies the port connecting the device to the PROFINET controller. Only necessary if topology is configured.
- **First device at port 2**
  Specifies the device that is configured as the first device in the network configuration as seen from the PROFINET controller. Only necessary if topology is configured.
- **First device port at port 2**
  Specifies the port connecting the device to the PROFINET controller. Only necessary if topology is configured.

A configuration example is available in Supported PROFINET IO configurations (Page 81).

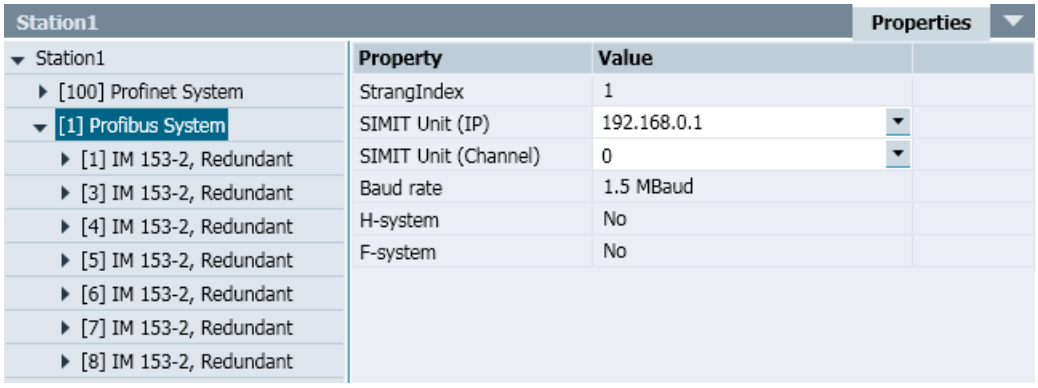
Properties of a PROFINET IO device:

The IP address and subnet mask of a PROFINET device are visible in the "Properties" window of the PROFINET device.

The IP address of an IO device can be edited. Editing of the IP address is necessary if the IP address is empty or multiple IO devices have the same IP address. Otherwise, the affected IO devices will not be simulated.

The subnet mask is editable in offline mode. The input of a subnet mask is checked for validity.

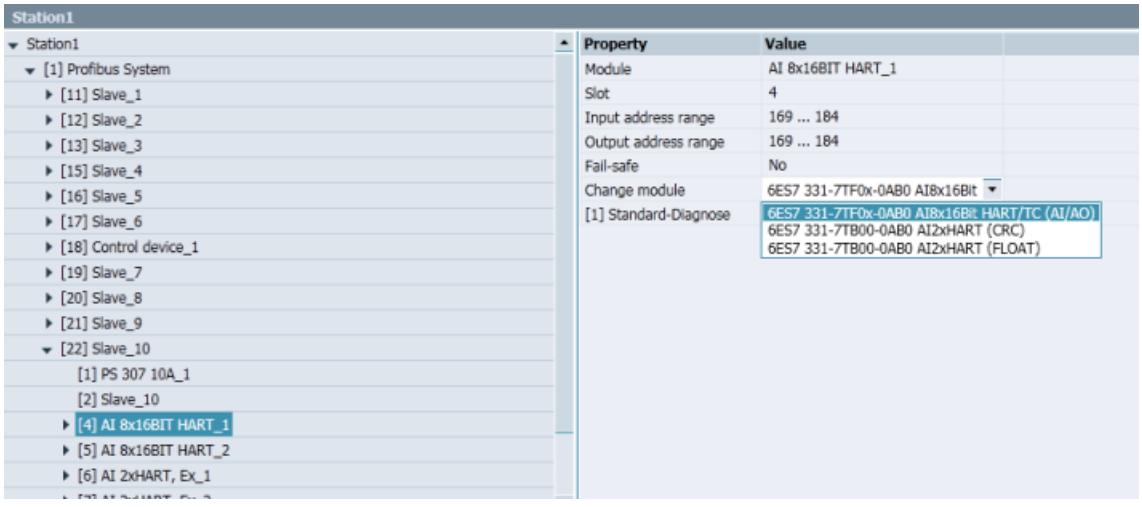
Specific properties of a connection over PROFIBUS



- **Baud rate**
  Shows the transmission speed.

Properties of a module

If different modules are available for a module type identifier, the user selects the right module in the "Properties" window of the module.



You will find a list of module types for which multiple modules are available for selection in the next section.

Modules that cannot be uniquely identified (Page 101)

Note

When the module is changed, the hardware configuration is reimported so that the associated signals are determined again. The import takes some time.

See also

Configuring the station (Page 107)

2.5.7.14 XML import interface for hardware configuration of third-party systems

Basics for XML import

SIMIT supports the import of the hardware configuration of controllers of third-party vendors via an XML hardware import interface.

An XML file that conforms to the formats described below can be imported into SIMIT (SU import / XML import). This creates an executable and loadable PROFIBUS or PROFINET project.

The user can thus configure a simulation relatively easily using a suitable export of the hardware data from his configuration tool or also using an XML editor.

XML format for the import of a PROFIBUS line

The following describes the structure of the XML file, which contains all the data required for SIMIT to simulate a PROFIBUS line.

`<?xml version="1.0" encoding="iso-8859-1"?>` `<HARDWARE>`	XML declaration line Root tag, (always "HARDWARE")
`<COMMENT/>`	Comment, optional
`<NAME>SimbaPROFIBUS_0[0]</NAME>`	Name of the simulation box, name is freely assignable, the number in the square brackets at the end indicates the channel of the box and must be present.
`<HWTYPE>DPBOX</HWTYPE>`	Type of simulation box, for PROFIBUS "DPBOX"
`<BAUDRATE>6</BAUDRATE>`	PROFIBUS baud rate: 9 = 12 MBd, 8 = 6 MBd, 7 = 3 MBd, 6 = 1.5 MBd, etc.
`<DPSUBSYSTEM>1</DPSUBSYSTEM>`	PROFIBUS system ID
`<CHILDTYPE>SLAVE</CHILDTYPE>`	Internal key for the next hierarchy level.
`<SLAVE>`	Tag for the PROFIBUS device
`<NAME>My ET200</NAME>`	Name (optional)
`<ORDERNUMBER>6ES7 326-1BK01-0AB0</ORDERNUMBER>`	Order number (optional)
`<INDEX>3</INDEX>`	PROFIBUS address
`<TYPE>801E</TYPE>`	Type identification according to GSD file
`<CHILDTYPE>MODULE</CHILDTYPE>`	Key for the next hierarchy level
`<REDUNDANCY>1</REDUNDANCY>`	Redundancy, only for redundant bus systems
`<MODULE>`	Tag for device modules
`<NAME>My DI16</NAME>`	Name, optional
`<INDEX>4</INDEX>`	Slot
`<TYPE>4301009FC2</TYPE>`	Module type according to GSD file

<CHILDTYPE>CHANNEL</CHILDTYPE>	Next hierarchy level
<INLEN>2</INLEN>	Input length of the module in bytes
<OUTLEN>0</OUTLEN>	Output length of the module in bytes
<LOGIN>0</LOGIN>	Logical input address, must be unique AS-wide. Data is exchanged from the interface via this address.
</MODULE>	
...	Additional modules can follow.
<MODULE>	
<COMMENT/>	
<NAME>DI 24xFailsafe </NAME>	
<ORDERNUMBER>6ES7 326-1BK01-0AB0</ORDERNUMBER>	
<INDEX>8</INDEX>	
<TYPE>C2838908C1</TYPE>	
<CHILDTYPE>CHANNEL</CHILDTYPE>	
<INLEN>10</INLEN>	
<OUTLEN>4</OUTLEN>	
<LOGIN>4</LOGIN>	
<LOGOUT>4</LOGOUT>	Logical output address, must be unique AS-wide. Data is exchanged from the interface via this address.
</MODULE>	
</SLAVE>	
<SLAVE>	Additional devices can follow
...	
</SLAVE>	
...	
</HARDWARE>	

XML format for the import of a PROFINET line

The XML file for importing a PROFINET line differs from that of a Profibus line due to the different bus hardware and the other associated parameters required.

<HARDWARE>	
<COMMENT/>	
<NAME>SimbaPNIO_2[0]</NAME>	Name of the simulation box, name is freely assignable, the number in the square brackets at the end indicates the channel of the box and must be present.
<HWTYPE>PNIO</HWTYPE>	Type of simulation box. For PROFINET "PNIO"
<IOSUBSYSTEM>100</IOSUBSYSTEM>	PROFINET system ID
<FIRSTDEVICE>6</FIRSTDEVICE>	Topology information, the device that is directly behind the controller and is thus simulated at the external interface of the simulation box.
<FIRSTPORT>1</FIRSTPORT>	The corresponding port for this.
<SECONDDEVICE>6</SECONDDEVICE>	Topology information for the second channel for redundant systems.

<SECONDPORT>2</SECONDPORT>	Second port for redundant systems.
<CHILDTYPE>PNIODEV</CHILDTYPE>	Child type of the next hierarchy level.
<PNIODEV>	Tag for a PROFINET device.
<NAME>My PROFINET IO</NAME>	Name of the device (optional).
<INDEX>6</INDEX>	Device index
<TYPE>002A0302</TYPE>	Device type according to GSDML
<IPADDR>192.168.0.4</IPADDR>	Simulated IP address.
<SubnetMask>255.255.255.0</ SubnetMask >	Subnet mask plus
<PRJNAME>im153-4pn</PRJNAME>	Device name in the project, required by the controller to identify the device using a DCP telegram.
<CHILDTYPE>PNIOSLOT</CHILDTYPE>	Type of the next hierarchy level.
<PNIOSLOT>	Tag for a PNIO slot.
<NAME>! IM153-4 PN HF V4.0</NAME>	Name (optional)
<ORDERNUMBER>6ES7 153-4BA00- 0XB0</ ORDERNUMBER>	Order number (optional), the order number can be useful if a distinction should be made between different modules with the same type identification.
<INDEX>0</INDEX>	Slot, 0 is always the head module with PROFINET. In contrast to PROFIBUS devices, this must be specified here.
<TYPE>00000423</TYPE>	Slot type according to GSDML
<API>0</API>	API no.
<HEAD>8</HEAD>	Identification of the slot as a head module, may only be present once in the device.
<CHILDTYPE>SUBSLOT</CHILDTYPE>	Type of the next hierarchy level
<SUBSLOT>	Tag for a PNIO subslot
<NAME>DIM 9 HF V4.0</NAME>	Name (optional)
<INDEX>1</INDEX>	Continuous index within the slot
<TYPE>00000000</TYPE>	Type according to GSDML
<INLEN>0</INLEN>	IO data input length in bytes
<OUTLEN>0</OUTLEN>	IO data output length in bytes
<IOPS_Length>1</IOPS_Length>	IO provider module status length in bytes
<IOCS_Length>1</IOCS_Length>	IO consumer module status length in bytes
<CHILDTYPE>CHANNEL</CHILDTYPE>	Type of the next hierarchy level
</SUBSLOT>	
<SUBSLOT>	
<NAME>PN-IO</NAME>	Subslot for the PNIO interface
<INDEX>32768</INDEX>	
<TYPE>00008002</TYPE>	
<INLEN>0</INLEN>	
<OUTLEN>0</OUTLEN>	
<IOPS_Length>1</IOPS_Length>	
<IOCS_Length>1</IOCS_Length>	
</SUBSLOT>	
<SUBSLOT>	
<NAME>Port 1</NAME>	Subslot for a port of the PNIO interface
<INDEX>32769</INDEX>	

<TYPE>0000C000</TYPE>	
<INLEN>0</INLEN>	
<OUTLEN>0</OUTLEN>	
<IOPS_Length>1</IOPS_Length>	
<IOCS_Length>1</IOCS_Length>	
</SUBSLOT>	
<SUBSLOT>	
<NAME>Port 2</NAME>	
<INDEX>32770</INDEX>	
<TYPE>0000C000</TYPE>	
<INLEN>0</INLEN>	
<OUTLEN>0</OUTLEN>	
<IOPS_Length>1</IOPS_Length>	
<IOCS_Length>1</IOCS_Length>	
</SUBSLOT>	
</PNIOSLOT>	
<PNIOSLOT>	Slot example for an input module
<NAME>SM 321 DI16xDC24V</NAME>	
<ORDERNUMBER>6ES7 321-1BH50-0AA0</ ORDERNUMBER>	
<INDEX>1</INDEX>	
<TYPE>00009FC2</TYPE>	
<API>0</API>	
<CHILDTYPE>SUBSLOT</CHILDTYPE>	
<SUBSLOT>	
<TYPE>00000000</TYPE>	
<INDEX>1</INDEX>	
<NAME>14</NAME>	
<INLEN>2</INLEN>	
<OUTLEN>0</OUTLEN>	
<IOPS_Length>1</IOPS_Length>	
<IOCS_Length>1</IOCS_Length>	
<LOGIN>6</LOGIN>	Logical input address, must be unique AS-wide. Data is exchanged from the interface via this address.
<CHILDTYPE>CHANNEL</CHILDTYPE>	
</SUBSLOT>	
</PNIOSLOT>	
...	Additional allocated slots
<PNIOSLOT>	
<NAME>FDO10xDC24V/2A</NAME>	
<INDEX>4</INDEX>	
<TYPE>000008C1</TYPE>	
<API>0</API>	
<CHILDTYPE>SUBSLOT</CHILDTYPE>	

<SUBSLOT>	
<TYPE>00000000</TYPE>	
<INDEX>1</INDEX>	
<INLEN>6</INLEN>	
OUTLEN>6</OUTLEN>	
<IOPS_Length>1</IOPS_Length>	
<IOCS_Length>1</IOCS_Length>	
<LOGIN>18</LOGIN>	
<LOGOUT>18</LOGOUT>	Logical output address, must be unique AS-wide. Data is exchanged from the interface via this address.
</SUBSLOT>	
</PNIOSLOT>	
</PNIODEV>	
...	Additional PNIO devices
<PNIODEV>	
...	
</PNIODEV>	
</HARDWARE>	

Simultaneous import of several lines

For the simultaneous import of several bus lines, you can arrange the individual "HARDWARE" tags within a "PROJECT" tag. This can be useful, for example, to generate the two bus lines of an H-system simultaneously.

<?xml version="1.0" encoding="iso-8859-1"?> <PROJECT>	XML declaration line Root tag, (now "PROJECT")
<HARDWARE>	first "HARDWARE"
...	
</HARDWARE>	
<HARDWARE>	second "HARDWARE"
...	
</HARDWARE>	
</PROJECT>	

Syntactic validation of generated XML files

The XML files generated for import in SIMIT can be checked for syntactic errors or missing keys before the import. A document type definition (DTD file) is available for this. However, the content of the tags is not checked for meaningfulness.

To validate your XML file, copy the following text block from the second line into your XML file and then load the entire file into a DTD validator.

```
<!ELEMENT HARDWARE (COMMENT?, NAME?, HWTYPE, ((DPSUBSYSTEM, BAUDRATE,
CHILDTYPE, SLAVE*) | (IOSUBSYSTEM, FIRSTDEVICE, FIRSTPORT, SECONDDEVICE?,
SECONDPORT?, CHILDTYPE, PNIODEV*)) )>
<!ELEMENT COMMENT (#PCDATA)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT HWTYPE (#PCDATA)>
<!ELEMENT CHILDTYPE (#PCDATA)>
<!ELEMENT BAUDRATE (#PCDATA)>
<!ELEMENT DPSUBSYSTEM (#PCDATA)>
<!ELEMENT IOSUBSYSTEM (#PCDATA)>
<!ELEMENT FIRSTDEVICE (#PCDATA)>
<!ELEMENT FIRSTPORT (#PCDATA)>
<!ELEMENT SECONDDEVICE (#PCDATA)>
<!ELEMENT SECONDPORT (#PCDATA)>
<!ELEMENT SLAVE (COMMENT?, NAME?, ORDERNUMBER?, INDEX, TYPE, CHILDTYPE,
REDUNDANCY?, MODULE*)>
<!ELEMENT INDEX (#PCDATA)>
<!ELEMENT TYPE (#PCDATA)>
<!ELEMENT ORDERNUMBER (#PCDATA)>
<!ELEMENT REDUNDANCY (#PCDATA)>
<!ELEMENT MODULE (COMMENT?, NAME?, ORDERNUMBER?, INDEX, TYPE, CHILDTYPE,
INLEN, OUTLEN, LOGIN?, LOGOUT?)>
<!ELEMENT INLEN (#PCDATA)>
<!ELEMENT OUTLEN (#PCDATA)>
<!ELEMENT LOGIN (#PCDATA)>
<!ELEMENT LOGOUT (#PCDATA)>
<!ELEMENT FAILSAFE (#PCDATA)>
<!ELEMENT PNIODEV (COMMENT?, NAME?, ORDERNUMBER?, INDEX, TYPE, IPADDR,
PRJNAME, CHILDTYPE, PNIOSLOT*) >
<!ELEMENT IPADDR (#PCDATA)>
<!ELEMENT PRJNAME (#PCDATA)>
<!ELEMENT PNIOSLOT (COMMENT?, NAME?, ORDERNUMBER?, INDEX, TYPE, API, HEAD?,
CHILDTYPE, SUBSLOT*) >
<!ELEMENT API (#PCDATA)>
<!ELEMENT HEAD (#PCDATA)>
<!ELEMENT SUBSLOT (NAME?, INDEX, TYPE, INLEN, OUTLEN, IOPS_Length,
IOCS_Length, LOGIN?, LOGOUT?, CHILDTYPE?) >
<!ELEMENT IOPS_Length (#PCDATA)>
<!ELEMENT IOCS_Length (#PCDATA)>
```

If you do not want to copy the text every time, copy the text block into a text file (for example, "SIMXMLImport.dtd"), and install only a link to this text file for the validation in your XML files in the second line.

```
<!DOCTYPE HARDWARE SYSTEM "SIMITXMLImport.dtd">
```

The first and last line of the text block must be omitted for use in an external DTD file.

XML validators on the Internet are, for example:

- https://www.xmlvalidation.com (https://www.xmlvalidation.com)
- https://www.w3schools.com/xml/xml_validator.asp (https://www.w3schools.com/xml/xml_validator.asp)

However, they only work if the DTD is directly integrated into the XML file.

The "XMLNotepad2007" from Microsoft is suitable for offline XML validation.

2.6 Virtual Controller

2.6.1 Operating principle of Virtual Controllers

2.6.1.1 Introduction

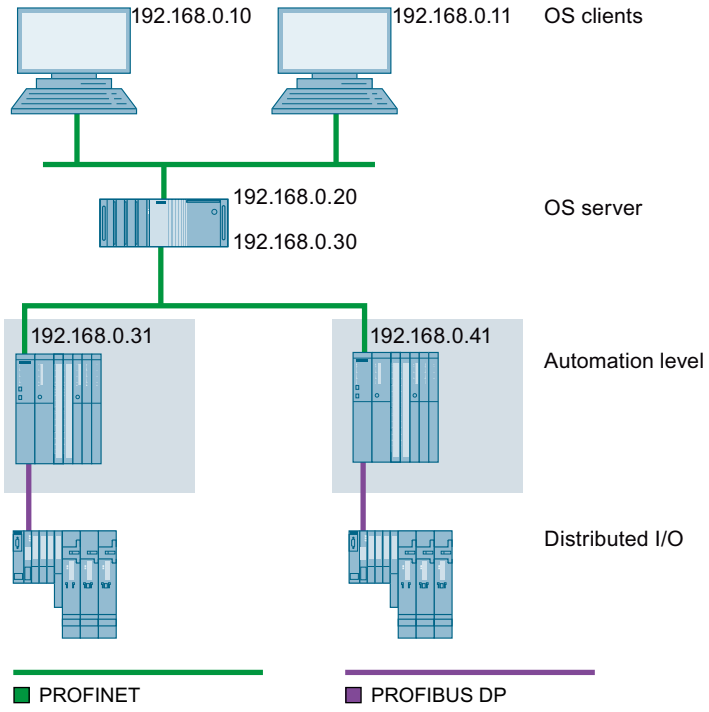
Function

The virtual controller simulates the response of a SIMATIC controller of the type S7-300 or S7-400. The virtual controller is loaded with the original PLC user program. The field device level and its connection over distributed bus systems are ignored. Instead, process simulation is connected directly over the process image input and output.

You can use up to 32 virtual controllers in the simulation project, and can distribute them across multiple PCs.

Virtual controllers can communicate with each other; connections to external partners and the operator control and monitoring system are also possible. Only IP-based connections in address space IPv4 are supported, however.

The figure below shows which automation level components are represented as virtual controllers:



In SIMIT, all virtual controllers in a simulation project are shown and configured as separate couplings in the project navigation under "Couplings > Virtual controllers". Data exchange between SIMIT and the virtual controller is over these couplings.

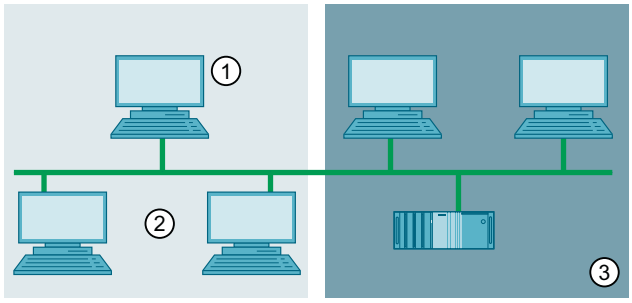
Intended use of a virtual controller

NOTICE
Important information: • Automation programs can be run on MS Windows with the virtual controller. As MS Windows is not a real-time operating system, the following limitations apply: – System availability – Accuracy of simulation You also cannot connect real I/O or bus systems. • While the functions of the virtual controller have been tested for its intended use (testing and training systems), we cannot guarantee that the system will always demonstrate the exact time response. The functions provided cannot cover the complete functional scope of a real controller in all situations. **The system is therefore not intended for the control of real plants.** **The results obtained for virtual commissioning must be checked and evaluated by the user.** To protect the confidentiality of your data, use the virtual controller only as part of a holistic Industrial Security concept that is state-of-the-art. Third-party products you are using should also be considered. You can find additional information on the Internet (https://www.siemens.com/industrialsecurity).

Network topology of simulation with a virtual controller

The entire simulation project including operator control and monitoring can run on one PC. If the PC does not have sufficient capacity, you can distribute virtual controllers across multiple PCs.

The figure below shows the network topology with virtual controllers distributed between multiple PCs:



TCP/IP

① PC with installed SIMIT Simulation Platform
- Manages the SIMIT project
- Starts/stops simulation
- Simulates the field and process level
- Can also include a virtual controller.

② PC with SIMIT VC (virtual controller) installed
- Hosts 1 to 8 virtual controllers
- Must be accessible in the network from the SIMIT PC
- Must be accessible from the OS server

③ PCs with PCS 7 installed (ES, OS server and client)

Note

Supported STEP 7 version

The virtual controller coupling is supported up to and including STEP 7 V5.5.

Note

Use of CFC charts on an S7-410 simulated in the virtual controller

If you are simulating an S7-410 with the virtual controller, note the following when using CFC charts:

Starting with CFC V8.2 Update 3, you can download a user program without any limitations. If you are using an older version of CFC, follow the stops below on each PC configured as a virtual controller:

1. Change to the following directory in Windows:
   "Programs (x86)\Siemens\Automation\SIMIT\SIMIT VC\data\CPU"
2. Rename the following files:
   – "6ES7 410 5HN08-0AB0" → "6ES7 410 5HN08-0AB0_CPU410"
   – "6ES7 410 5HX08-0AB0" → "6ES7 410 5HX08-0AB0_CPU410"
3. Rename the following files:
   – "6ES7 410 5HN08-0AB0_LEGACY" → "6ES7 410 5HN08-0AB0"
   – "6ES7 410 5HX08-0AB0_LEGACY" → "6ES7 410 5HX08-0AB0"

As a result of this action, the virtual controller will signal back as S7-417. You can download the project with CFC charts from PCS 7 to the PLC. If you update your CFC installation to the latest version, you must undo this renaming.

Note

Delta download

With each delta download, SIMIT saves a time stamp in the controller. Based on the time stamp, a decision is made as to whether the program is identical in the controller ("online") and in SIMIT ("offline"). The virtual controller does not support saving this time stamp across a restart of the simulation but stores the program correctly. SIMIT will therefore signal an inconsistent status of the program when going online after a delta download and restart of the simulation. You can avoid this message with a complete download.

Note

BlockPrivacy

The virtual controller cannot execute blocks that are protected with "BlockPrivacy".
If your project contains blocks that are protected this way, these are replaced by a NOP (No Operation) in the virtual controller.

2.6.1.2 Requirements for a simulation network

Avoiding IP address conflicts

When using a virtual controller, all of its configured IP addresses and the connections based on them are applied. You therefore need to enter these additional IP addresses in the network settings of the PC.

NOTICE
Adding and changing IP addresses Changing an IP address can lead to communication problems in the network. For example, it may no longer be possible to access a PC. Never change IP addresses without consulting your IT administrator.

Permitted network configurations

A PC connected to the simulation network can be located in multiple networks so long as there are no subnet overlaps between the networks. Access to the different networks can be provided either by multiple network adapters or by configuring multiple IP addresses from different subnets on one network adapter.

Impermissible subnet overlaps occur when the PC has access to the same network or subnet via more than one IP address, e.g.:

- 192.168.1.1/16
- 192.168.2.1/24

In this case, the correct configuration for clearly separating the PC's access to both subnets is as follows:

- 192.168.1.1/24
- 192.168.2.1/24

Subnet overlaps can result in faults when configuring additional PCs in the distribution editor and during simulation operation.

Rules for simulation with a Virtual Controller

Observe the following rules:

- If a virtual controller is to establish a connection for operator control and monitoring, it must be reachable over the plant bus.
- The PC on which SIMIT Simulation Platform is installed must have its IP address in the same subnet as the PCs on which a virtual controller is running.
- Each IP address is unique.
- Each computer name is unique.
- The IP addresses of the simulated stations are entered in the network settings of the PCs.

Note

Power saving options

After installation of SIMIT SP or SIMIT VC, disable the power saving options of the network adapters and the system on all computers. This will prevent shutdown of network and system functions by the operating system in simulation mode.

Detailed information is available in the documentation of your operating system.

2.6.1.3 Functions

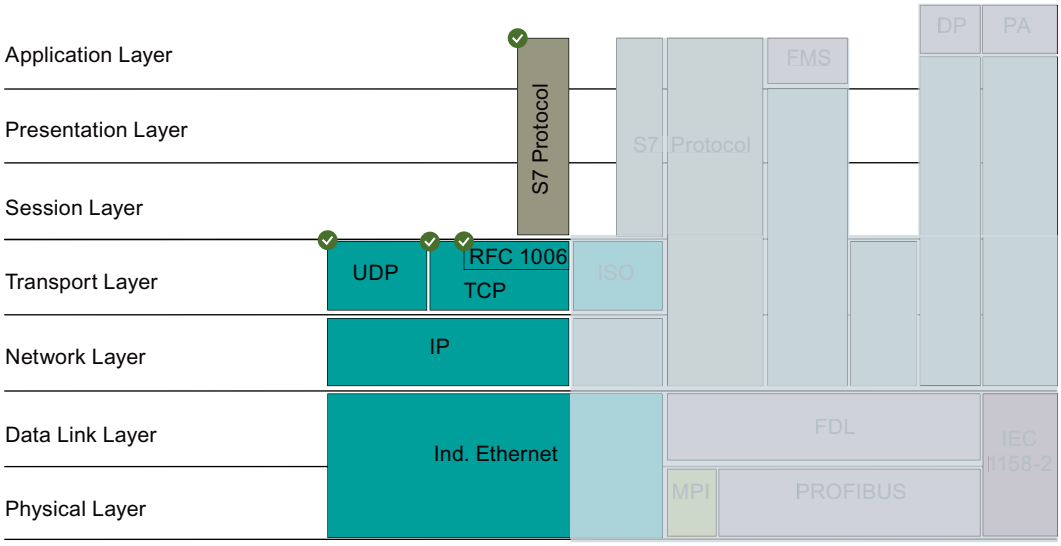
The virtual controllers support the following basic controller functions, irrespective of the specific type:

- Downloading the hardware configuration and application software to the virtual memory of the virtual controllers
- Interpretation of MC7 code
- Provision of data areas, arithmetic unit, batteries and results of logic operations, counters, timers, memory areas for process images, etc.
- OB management and scheduling
- Communication functions
- A selection of system functions (SFB/SFC) insofar as this is possible and useful in virtualization.

2.6.1.4 Basics of Virtual Controller communication

Introduction

The virtual controller supports IP-based communication over Ethernet / PROFINET and the following protocols:



Supported:

- S7
- S7H
- TCP
- UDP
- RFC 1006

The simulation of Ethernet CPs is limited to IP communication. Other CPs are not simulated. The module status of CPs is not displayed.

The system differentiates between communication between the following stations:

- Virtual Controller ⇔ External partner
- Virtual Controller ⇔ Virtual Controller within a SIMIT instance

Connection configuration in SIMIT

The virtual controller reads the connection configurations from the downloaded SDBs (system data blocks). Configuration changes to a connection are therefore only possible in the STEP 7/ PCS 7 project. The changes must be made known to the virtual controller by download.

You can individually enable and disable the use of IP addresses of a virtual controller in the Distribution Editor.

Communication with an external partner

The following stations are defines as "external partners":

- Any other device, e.g. a real PLC
- OS servers, e.g. WinCC
- Virtual Controller that runs in the context of another SIMIT instance (e.g. with distributed simulation)

Note

Please note the restrictions that can result when using substitute functions in virtual controllers.

A connection to the virtual controller is only established for fully qualified connections if both partners actually use the respective specified endpoint parameters at runtime (IP address, TSAP or TCP/UDP port number). For example, WinCC always uses the primary IP address. This is problematic if multiple IP addresses have been configured on a network adapter and the connection configuration does not exclusively refer to the primary IP address.

We therefore recommend that you set up WinCC on computers with only one IP address to avoid problems when connecting to the virtual controller.

Communication between two virtual controllers within a SIMIT instance

All IP-based connections between two virtual controllers are supported.

Open User Communication (TSEND / TRECV)

Within the framework of Open User Communication (TSEND / TRECV), the IP addresses and ports that are stored in the data blocks of the S7 program are used by default. These IP addresses must be entered in the network settings of the PCs and the port numbers must be available.

If you are using Open User Communication exclusively between virtual controllers, you can use simulation-internal addressing.

You do not have to enter the IP addresses stored in the S7 program in the network settings of the PCs in this case. Conflicts with existing port numbers are avoided. You define this type of addressing once for the entire project and for each protocol separately in the Distribution Editor (Page 144).

See also

Supported system functions (Page 130)

Supported S7 blocks (Page 131)

Supported services (Page 132)

2.6.1.5 Supported system functions

Unlike a real controller, the virtual controller runs on PC hardware and on a Windows operating system. It cannot therefore be expected to simulate all properties, functions and responses of a real controller.

Only the standard functionalities, for example those of an S7-417-5H, are supported. For extensions and additional functions of an AS 410-5H, virtual controllers map the functionality of an S7-400 CPU 417-5H.

Substitute functions in virtual controllers

Automation programs consist of MC7 code that can be interpreted by the virtual controller, and system functions (SFC/SFB) that are part of the firmware of a controller and the original of which cannot be run on the PC. The virtual controller has a modified substitute implementation for the most important system functions:

Type	Block number	Comment
SFB	0–5, 8, 9, 12–15, 22, 23, 31, 33–36, 37[1]	
SFB	52, 53, 54	
SFC	0	The virtual controllers are internally always synchronized with the computer time; SET_CLK therefore only has a temporary effect
SFC	1–6, 8, 9[1], 10[1], 13–15, 17–34, 36–44, 46–50	
SFC	51	SZLs are supported by the virtual controller with restrictions as there is no simulation of the distributed I/O
SFC	52, 58, 59, 70, 71, 78, 80, 81, 82, 83, 84, 108, 109	
SFC	62, 64, 79, 80, 81, 85[2], 87, 90, 107, 115[3], 128[3], 129[3]	
SFC	131, 132, 133, 134, 135, 136	T communication

[1] Only implemented as NOP

[2] Only for S7-300

[3] Only for S7-410

Note

- The use of SFCs is limited because, for example, the I/O are not simulated.
- Before using virtual controllers, verify that the SFBs and SFCs required by your automation system are supported. You can check the reference data of the blocks in SIMATIC Manager, for example. You can find additional information on the reference data in the STEP 7 help.
- The substitute functions provide useful feedback on standard operation of the hardware without faults, provided this is possible with the level of implementation and information from the SIMATIC project. Only very limited information on response in the event of a fault is possible, and must be checked by the user independently.
- User blocks and library components (FB, FC) can potentially access system functions that are not supported by the virtual controller. In this case, the entire block is ignored (NoOperation).
- If the automation program expects signals from these system functions, you can make the corresponding (instance) DB in SIMIT addressable and write the expected information for SIMIT straight to the corresponding DB.
- The virtual controller supports the "S7 F Systems Lib V1_3" library and "Safety Matrix" in failsafe operation. Error states cannot be mapped. Only limited use of snapshots is possible in connection with the "S7 F Systems Lib".
- "Distributed Safety" is not supported.

2.6.1.6 Supported S7 blocks

The following table shows the S7 blocks supported for each protocol used and communication partner:

Communication between:	Virtual controller-Virtual controller				Virtual controller-External partner			
S7 block	S7	TCP	UDP	ISO-on-TCP	S7	TCP	UDP	ISO-on-TCP
FB12 BSEND	x	–	–	–	x	–	–	–
FB13 BRCV	x	–	–	–	x	–	–	–
FB14 GET	x	–	–	–	x	–	–	–
FB15 PUT	x	–	–	–	x	–	–	–
FB8 USEND	x	–	–	–	x	–	–	–
FB9 URCV	x	–	–	–	x	–	–	–
FC5 AG_SEND	–	x	x	x	–	x	x	x
FC50 AG_LSEND	–	x	x	x	–	x	x	x
FC53 AG_SSEND	–	x	x	x	–	x	x	x
FC6 AG_RECV	–	x	x	x	–	x	x	x
FC60 AG_LRECV	–	x	x	x	–	x	x	x
FC63 AG_SRECV	–	x	x	x	–	x	x	x
SFB12 BSEND	x	–	–	–	x	–	–	–
SFB13 BRCV	x	–	–	–	x	–	–	–
SFB14 GET	x	–	–	–	x	–	–	–
SFB15 PUT	x	–	–	–	x	–	–	–

Communication between:	Virtual controller-Virtual controller				Virtual controller-External partner			
S7 block	**S7**	**TCP**	**UDP**	**ISO-on-TCP**	**S7**	**TCP**	**UDP**	**ISO-on-TCP**
SFB22 STATUS	x	–	–	–	x	–	–	–
SFB23 USTATUS	x	–	–	–	x	–	–	–
SFB31 NOTIFY_8P	–	–	–	–	x	–	–	–
SFB33 ALARM	–	–	–	–	x	–	–	–
SFB34 ALARM_8	–	–	–	–	x	–	–	–
SFB35 ALARM_8P	–	–	–	–	x	–	–	–
SFB36 NOTIFY	–	–	–	–	x	–	–	–
SFB8 USEND	x	–	–	–	x	–	–	–
SFB9 URCV	x	–	–	–	x	–	–	–
SFC107 ALARM_DQ	–	–	–	–	x	–	–	–
SFC108 ALARM_D	–	–	–	–	x	–	–	–
SFC17 ALARM_SQ	–	–	–	–	x	–	–	–
SFC18 ALARM_S	–	–	–	–	x	–	–	–
SFC19 ALARM_SC	–	–	–	–	x	–	–	–
SFC131 T_SEND	–	x	–	x	–	x	–	x
SFC132 T_RCV	–	x	–	x	–	x	–	x
SFC133 T_CON	–	x	x	x	–	x	x	x
SFC134 T_DISCON	–	x	x	x	–	x	x	x
SFC135 T_USEND	–	–	x	–	–	–	x	–
SFC136 T_URCV	–	–	x	–	–	–	x	–

2.6.1.7 Supported services

Introduction

Real controllers provide comprehensive services that allow access to the controller.

SIMIT VC supports the following communication services, which are required for communication between virtual controllers and PCS 7 OS/WinCC servers or PCS 7 OS/WinCC clients. These services provide answers to the following queries that can be sent over the S7_DOS interface of PCS 7 OS/WinCC.

Note

The virtual controller does not answer other queries. Systems that rely on other services cannot therefore communicate correctly with the controllers emulated. These include, for example:

- BRAUMAT Classic
- AS based Batch

Supported PCS 7 OS / WinCC services

- VFD / Virtual Device Services
  - Read SZL (only the scope of SZL IDs that is required for PCS 7 OS/WinCC communication is implemented)
- OCM Services
  - Cyclic Read Variables (Start, Stop, Change, Abort, Delete)
  - Read Variables
  - Write Variables
- Message Services
  - Acknowledge
  - Announcement for Messages (not SCAN, LT group messages or archive)
  - Lock / Unlock Messages (not SCAN, LT group messages or archive)
  - Message Update (not SCAN, LT group messages or archive)
- PBK Services
  - USEND / URCV
  - BSEND / BRCV
  - PUT (Write Variables)
  - GET (Read Variables)

Supported S7 services

- SPS7
  - Set up application relationship
  - Cancel application relationship
  - Virtual device status
  - DataExchange2 Cancel
- Operator control and monitoring
  - Reading (variable / segmented)
  - Writing (variable / segmented)
  - Set up cyclic reading
  - Set up change-triggered cyclic reading
  - Start job
  - Stop job
  - Delete job
  - Change cyclic reading without delta evaluation
  - Change cyclic reading with delta evaluation

- Diagnostics
  - Log on/off
  - Acknowledgment
  - Acknowledgment to all
  - Lock messages
  - Release messages
  - Lock to all
  - Release to all
  - Update
  - Request update to all
  - Read SZL (forward to SoftPLC online interface)
- Programmable block communication
  - ALARM MELD
  - ALARM 8 MELD
  - ALARM SQ / ALARM DQ
  - ALARM S / ALARM D
  - ALARM MELD QTM
  - ALARM MELD 8 QTM
  - NOTIFY 8 ALARM MELD
  - USEND / URCV
  - BSEND / BRCV
  - GET
  - PUT
- Test and commissioning / object management
  - Forwarding to online interface

Supported services for a programming device (PG)

- Loading
- Delta loading
- Online monitoring (values)

- Read diagnostic buffer
- Type Change in Run (TCiiR)

Note

SIMIT VC supports type update with interface changes in RUN mode (Type Change in RUN (TCiR)).
This makes it possible to update the instances and download them to the target system in RUN mode after changing an interface at block types.

Note

Forcing is not supported.

2.6.1.8 Handling sync errors

Not all sync errors are detected or correctly dealt with (call of an error OB). The error may not be detected, but generate a runtime error (RTE).

Overview of virtual controller error handling:

Error description	Supported
BCD conversion error	Yes
Range length error when reading	No, generates RTE
Range length error when writing	No, generates RTE
Range error when reading	Yes
Range error when writing	Yes
Timer number error	No, generates RTE
Counter number error	No, generates RTE
Alignment error when reading	Yes
Alignment error when writing	Yes
Data block write error	No, writing to write-protected block
Instance data block write error	No, writing to write-protected block
Block number error DB	No, generates RTE
Block number error DI	No, generates RTE
Block number error FC	No, generates RTE
Block number error FB	No, generates RTE
DB not loaded	Yes
FC not loaded	Yes
FB not loaded	No, generates RTE

2.6.1.9 Sequential control

The following organization blocks of a real controller are supported:

Supported by SIMIT VC	Block type	Block	Description and event	Priority
1	User program	OB1	User program processed after hot restart (end of OB100) and at the end of the cycle.	1
10–17	Time-of-day interrupt	OB10 OB11 OB12 OB13 OB14 OB15 OB16 OB17	Time and date. A trigger can be initiated at a specific point in time so that a time OB (10–17) is called which then executes a program. Also called time-of-day interrupt OB.	2
20–23	Time-delay interrupt	OB20 OB21 OB22 OB23	Time-delay interrupts. After a delay time, the OB (20–23) is called and the program is executed.	3 4 5 6
30–38	Cyclic interrupt	OB30 OB31 OB32 OB33 OB34 OB35 OB36 OB37 OB38	Cyclic interrupts, start periodically after a defined time. Similar to clock signals, but much more accurate. Processing of OB1 is interrupted as they have a higher priority.	7 8 9 10 11 12 13 14 15
–	Hardware interrupt	OB40 OB41 OB42 OB43 OB44 OB45 OB46 OB47	Hardware interrupts. Respond at interruptible input, output or function modules to configured events, for example, positive edge, high limit violation. Are used, for example, when the response time in the program is too long.	16 17 18 19 20 21 22 23
–	DPV1 interrupt	OB55 OB56 OB57	DPV1 interrupts. Status, update or manufacturer-specific interrupts are triggered with DPV1 slaves.	2
–	Multicomputing interrupt	OB60	Synchronous operation of multiple CPUs	25
–	Synchronous cycle interrupt	OB61 OB62 OB63 OB64	Configuring short and equal process response times at the PROFIBUS DP	25

–	Background cycle	OB90	For program execution in the background	29
100–102	Startup	OB100 OB101 OB102	After a CPU restart (warm start) After a CPU hot restart After a CPU cold restart	27
121, not 122	Synchronous errors	OB121 OB122	When a module fault occurs	29

The sequential control system of SIMIT does not interrupt OBs already running. They are classified according to their priority in the basic cycle of SIMIT .
The following figure has a basic cycle time of 100 ms:

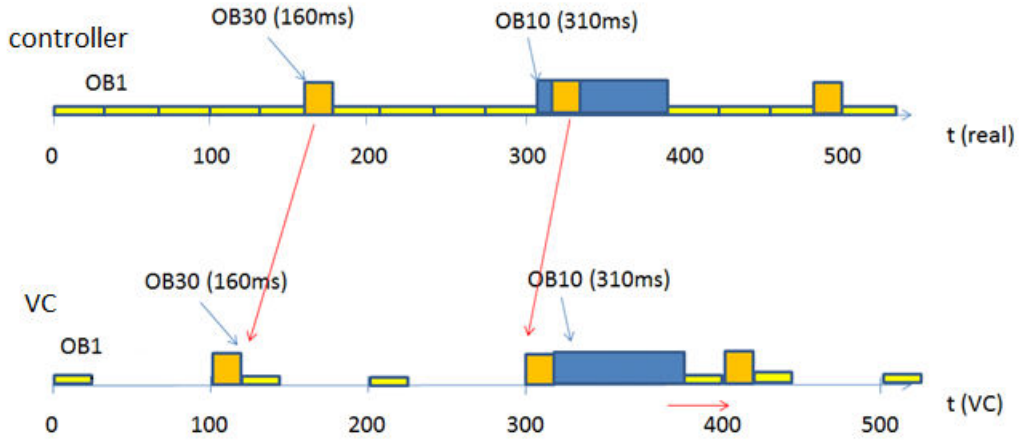


Figure 2-3 Sequential control

Resultant time response:

In each cycle, the pending OBs are executed in order of priority one after another without delays. In the figure above, the cycle time is exceeded for the value "400". This means that the total cycle increases and the next basic cycle is therefore delayed. In the subsequent cycle, SIMIT internally corrects all times (as shown in the figure above), changing them from "400+x" back to "400", and continues calculations on that basis even if the real time is already at "400+x". This keeps the virtual simulation time consistent.

Note

SIMIT distinguishes between timer times and the system time, which is used, for example, to send messages. The system time is regularly synchronized with the system time of the SIMIT computer, irrespective of the progress of the timer times.

2.6.1.10 Data record communication

The virtual controller supports data record communication over the following SFCs:

- WR_REC (SFC 58)
- RD_REC (SFC 59)

- WRREC (SFB 53)
- RDREC (SFB 52)
- DPNRM_DG (SFC13)

These blocks need a corresponding device model in SIMIT that can provide the data records.

See also

Access to a data record or memory area (Page 216)

2.6.1.11 Substitute mechanism for data exchange

Signals connected over PROFIBUS or PROFINET are not directly available through the process image. All other signal connections, such as Modbus or FF, need a substitute mechanism.

Note

You can enter substitute mechanisms on the "DB" tab in the coupling editor of the virtual controller coupling. You can find additional information in section: Properties of the Virtual Controller coupling (Page 152).

See also

Supported system functions (Page 130)

2.6.1.12 Notes on use of a virtual controller

Creating snapshots

The virtual controller can save snapshots while simulation is running. The snapshots are saved in the SIMIT project directory and are therefore part of a SIMIT project archive. Snapshots remain unchanged even if changes are made to the SIMIT project.

The following information is not included in snapshots:

- The states of connected systems
- Communication between virtual controllers
- Communication between virtual controllers and connected systems

Note

Snapshots are assigned to a virtual controller on the basis of its name.

If you change the name of a virtual controller in the project navigation after a snapshot is created, the snapshot can no longer be linked to that virtual controller. Do not rename the virtual controller in the project tree after creating a snapshot.

Note

Snapshots after changes to simulation

Snapshots save the states of the model and of individual virtual controllers.

- Please note that data in the snapshot is mapped by name (signal names, DB numbers, etc.). There is therefore no guarantee that a snapshot will lead to a useful simulation or virtual controller state after changes to the simulation (model or automation program). It is irrelevant in this case whether the change was caused by the user or the user program (for example, by using system functions such as CREAT_DB).
- Please note in particular that any existing snapshots may no longer be correctly assigned if a virtual controller is renamed.

2.6.2 Configuring the Virtual Controllers

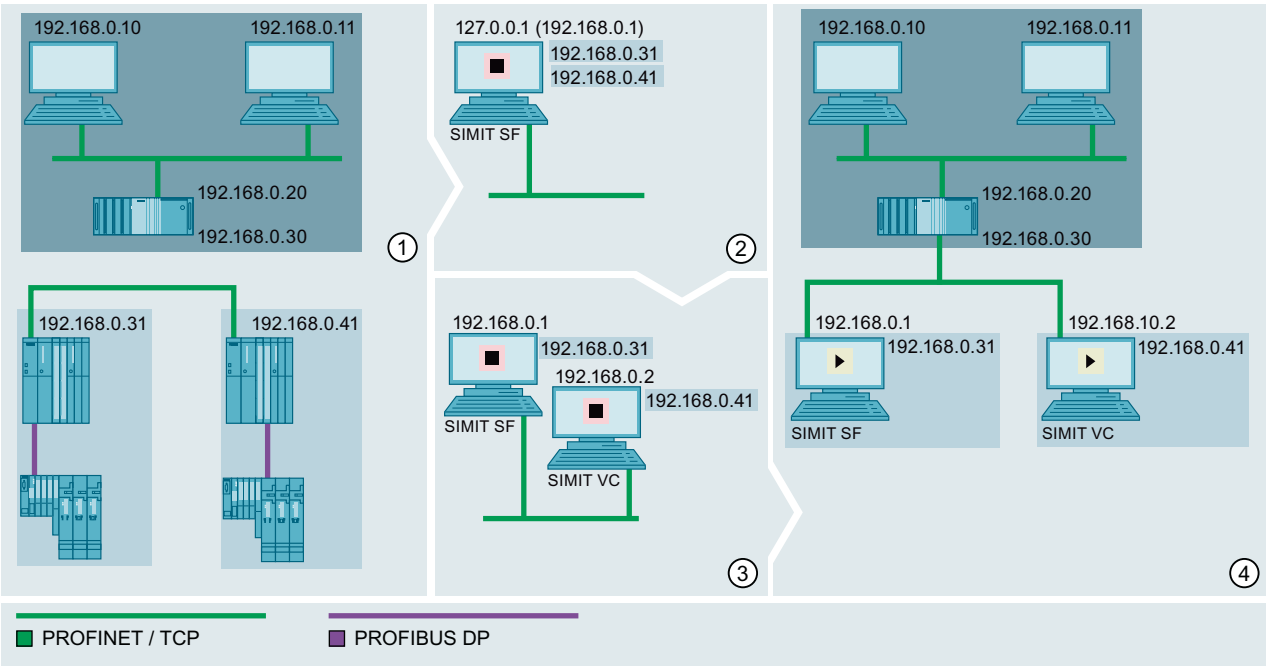
2.6.2.1 Configuring virtual controllers

Requirement

- The IP addresses of the stations on the plant bus and terminal bus are known.
- The IP addresses in the company/simulation network are known.
- All IP addresses are unique.

Action overview

The figure below shows the procedure for creating a Virtual Controller coupling:



Automation level with ES, OS server and two OS clients

Control level with PLC and distributed IO.

■ Simulation in "Stop" state

▶ Simulation in "Started" state

① Preparing the simulation

1. Define stations to be simulated.
2. If necessary, disconnect the control level from the network.

② Creating a SIMIT project:

1. Create a virtual controller coupling.
2. Import stations as virtual controllers from STEP 7/PCS 7 project.
   The SIMIT PC is assigned the IP address 127.0.0.1 in the SIMIT project. In the network, the SIMIT PC can still be accessed at its original IP address, e.g. 192.168.0.1.
3. Enter the IP addresses of the virtual controllers in the network settings of the SIMIT PC.
4. Create a simulation model.

③ Set up an additional PC (optional)

1. Change the IP address of the SIMIT PC in the distribution editor to the IP address visible in the network.
2. Integrate the PC into the network.
3. Distribute virtual controllers between the PCs.
4. Adapt the IP addresses in the network settings of the PCs.

④ Start simulation.

Note

When licenses are missing, the simulation with virtual controllers is set to "Stop" state. The simulation is not exited in this case.

See also

Creating a Virtual Controller (Page 141)

HWCNExport introduction (Page 62)

Importing a STEP 7/PCS 7 project (Page 142)

Configuring an additional PC (Page 149)

Properties of the Virtual Controller coupling (Page 152)

2.6.2.2 Creating a Virtual Controller

Introduction

Create a virtual controller coupling in a SIMIT project. Under a virtual controller coupling, you can import stations from one or more STEP 7/PCS 7 projects.

Note

Supported STEP 7 version

The virtual controller coupling is supported up to and including STEP 7 V5.5.

Requirement

- Project is open in SIMIT.
- No virtual controller coupling has been saved in the project.
- There is a STEP 7/PCS 7 project or HWCNExport file.

Procedure

Proceed as follows to create a virtual controller coupling:

1. Select the "New coupling" command from the "Couplings" shortcut menu in the project tree. The "Selection" dialog box opens.
2. Select the coupling type "Virtual controller". The "Virtual controller import" dialog box opens.
3. Configure station import from a STEP 7/PCS 7 project or the HWCNExport file.

Note

After creating a virtual controller, the complete station must be downloaded to the running virtual controller because the virtual controller reads information from the system data block during runtime.

When you make changes to the controller configuration, the complete station must once again be downloaded.

Result

You can find the result after the import of a STEP 7/PCS 7 project under Importing a STEP 7/PCS 7 project (Page 142).

If you complete the "Virtual controller" type coupling without importing a STEP 7/PCS 7 project, only the "Virtual controller" folder is created under "Couplings" in the project navigation. An empty OB1 is automatically generated in this case.

See also

"Virtual controller import" dialog box (Page 985)

"Selection" dialog box (Page 999)

Configuring virtual controllers (Page 139)

2.6.2.3 Importing a STEP 7/PCS 7 project

Introduction

To allow you to create a simulation with a virtual controller coupling, import the stations from a STEP 7/PCS 7 project. The following information from the STEP 7/PCS 7 project is applied:

- CPU
- Central I/O modules
- Distributed I/O modules
- Connections

You can run the import again at any time, for example to update configuration data or to add new stations. Identical stations are identified using the following criteria:

- Project name
- Station name
- Storage location in the STEP 7/PCS 7 project

As an option, you can also select the stations to be imported when you configure import.

Note

Supported STEP 7 version

The virtual controller coupling is supported up to and including STEP 7 V5.5.

Requirement

- A "Virtual Controller" type coupling has been created.
- SIMATIC Manager is installed or there is a HWCNExport file.
  Creating a HWCNExport file from STEP 7 projects (Page 63)

Procedure

Proceed as follows to import a STEP 7/PCS 7 project:

1. Select the "Import" command in the "Couplings > Virtual controller" shortcut menu in the project navigation.
   The "Virtual controller import" dialog box opens.
2. Select the required STEP 7/PCS 7 project or the HWCNExport file.
3. Select the mode for importing the symbols.
4. Select the required stations.
5. Click "Import".

Result

A Virtual Controller is created in the project navigation under "Couplings > Virtual controller" for each station from the STEP 7/PCS 7 project. The Virtual Controllers imported are assigned to the emulation PC "My Computer" in the "Distribution" editor after the first import.

See also

HWCNExport introduction (Page 62)

"Virtual controller import" dialog box (Page 985)

Configuring virtual controllers (Page 139)

Editing the signals of a coupling (Page 210)

2.6.2.4 Configuring virtual controllers

Requirement

- The STEP 7/PCS 7 project has been imported.
- A Virtual Controller coupling has been created.

Procedure

Proceed as follows to configure a Virtual Controller:

1. Double-click on the required virtual controller under "Couplings > Virtual Controller" in the project navigation.
2. Configure the virtual controller in the property view:
   - Select the "Time slice".
   - If necessary, enter the serial number of the "CPU" and/or memory card.
   - Adjust the cycle time of the OBs if necessary.
3. Import selected address ranges of one or more data blocks as required.

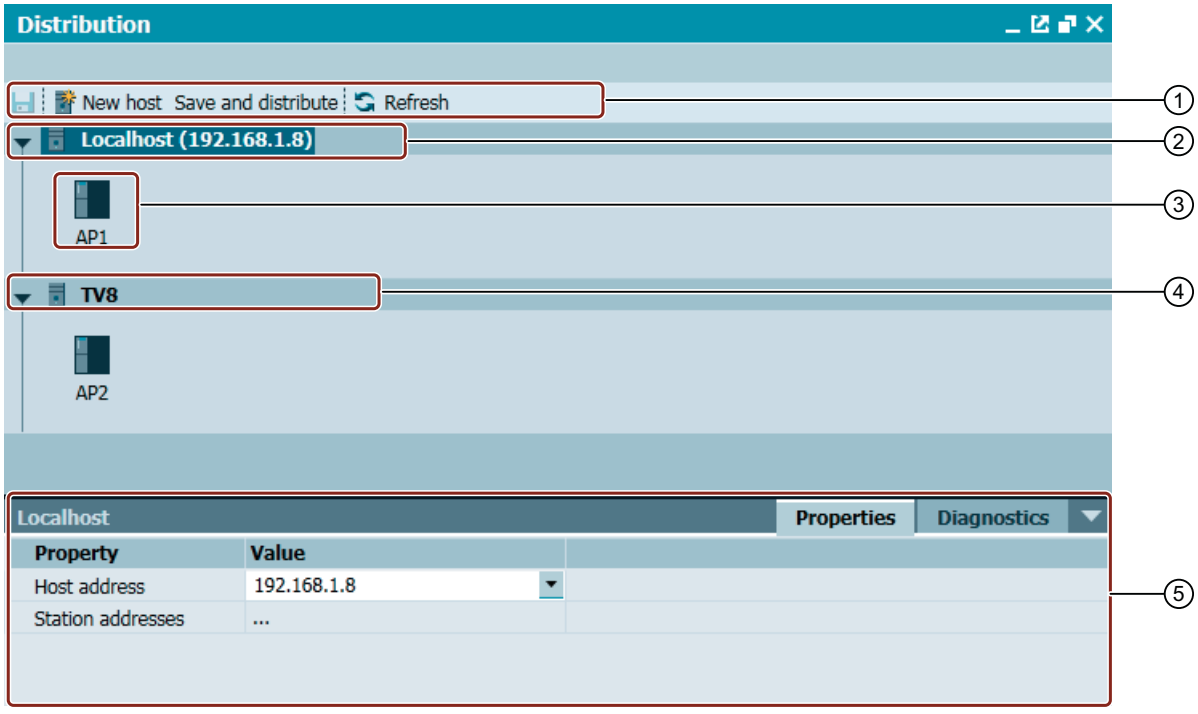
2.6.2.5 Distribution editor (Virtual Controller)

Distribution editor

In the distribution editor, you distribute the virtual controllers between additional PCs. This spreads the computing load for simulation across multiple PCs.

Double-click to open the distribution editor under "Couplings > Virtual controller" in the project navigation.

The figure below shows the structure of the distribution editor:



① Menu bar

② Represents the PC on which the SIMIT project is open.

③ Virtual controllers that you have imported from a STEP 7/PCS 7 project. These virtual controllers are by default assigned to the PC "My Computer".

④ Additional PC (optional)

The PC must be located in the same network as the PC "My Computer". You distribute the available virtual controllers among the PCs with drag & drop.

⑤ Property view

In the property view, you configure the properties of the selected PC or virtual controller. Conflicts such as duplicate IP addresses and inaccessible PCs are also displayed under "Diagnostics".

Menu bar

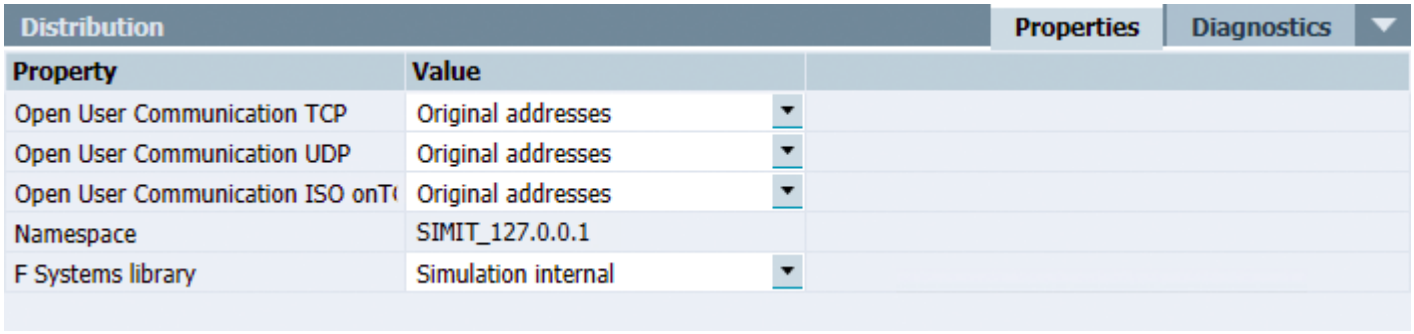
You can execute the following functions in the menu bar of the distribution editor:

- "Save"
  Saves the configuration.
- "New host"
  Adds a new PC. The PC must be located in the same network as the PC "My Computer".

- "Save and distribute"
  Saves the configuration and distributes the simulation project across the configured PCs.
- "Refresh"
  Refreshes the view:
  - Checks availability of configured PCs.
  - IP addresses are checked for conflicts. Conflicts are displayed in the property view under "Diagnostics".
  - If you remove a PC, its virtual controller is automatically moved to the PC "My Computer".

Properties of "Distribution"

Double-click on "Distribution" in the project tree to open the "Distribution" properties.



It specifies for each protocol which addresses are used for the identification of the communication partners. You can find additional information in section Basics of Virtual Controller communication (Page 128).

Restrictions in the simulation of or with fail-safe:

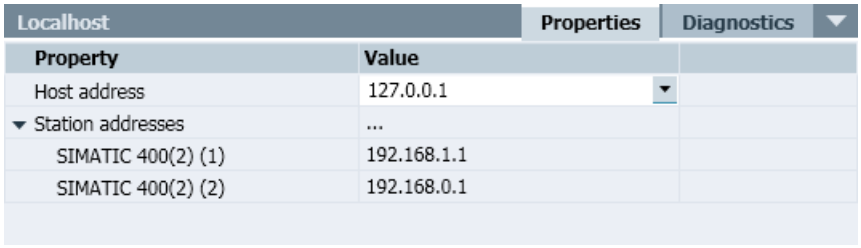
The simulation of the F Systems Library is made with block substitutions by default, which brings the following restrictions:

- Fail-safe communication to real controllers is not possible.
- Simplest simulation of the F Systems Library
- No mapping of error conditions

Alternatively, it is possible to switch to execution of the original blocks. This brings the following restrictions:

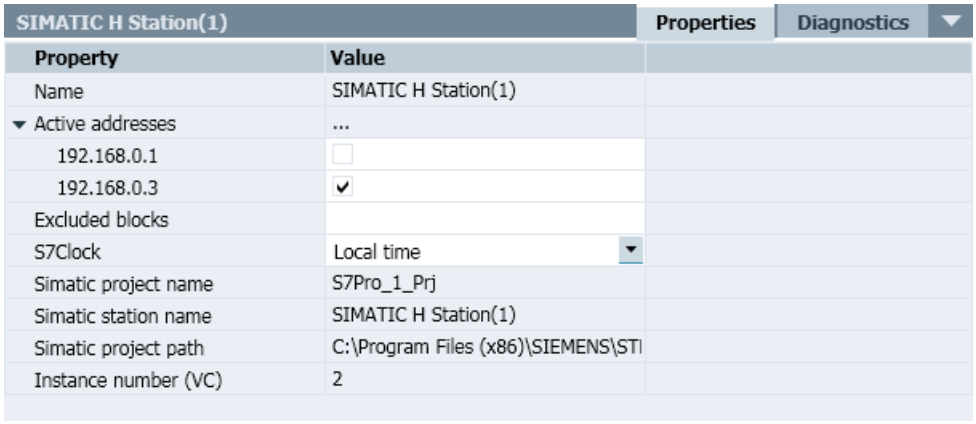
- Snapshots can only be used to a limited extent.
- After importing a snapshot, the F-program must be restarted.
- Increased performance requirements when executing original F-blocks

Properties of "My Computer" / "Emulation PC"



- "Computer name"
  Specifies the computer name of the PC on which a virtual controller is to be hosted.
  All PCs including "My PC" must be in the same subnet. As long as the PC "My PC" is set on the LoopBack adapter (127.0.0.1), no other PCs are shown.
- "Computer address"
  Shows the IP address of the PC by which it is currently identified in the network.
- "Station addresses"
  Shows the IP addresses of the virtual controllers that are assigned to this PC.
  Enter these IP addresses in the network settings of this PC. If an IP address is not to be used in the simulation, disable that IP address in the virtual controller settings.

Properties of a virtual controller



- "Name"
  Shows the name of the virtual controller.
  The name is taken from the STEP 7/PCS 7 project.
- "Active IP addresses"
  Shows the configured IP addresses of the virtual controller.
  Lists the IP addresses of the PLC and CPs plugged into the station. Activate the corresponding IP address if you want to download the virtual controller via this address.
- "Excluded blocks"
  Specifies the STEP 7 blocks that are not to be included in the simulation.
  Separate multiple blocks with commas. Invalid entries are deleted automatically when you exit the text box.

- "S7 Clock"
  Specifies the time format for the internal clock supplied to the virtual controller by the PC.
  - Local time
  - Universal Time Coordinated
- "Project name (SIMATIC)"
  Shows the name of the STEP 7/PCS 7 project.
- "Station name (SIMATIC)"
  Shows the station name.
- "Storage path (SIMATIC)"
  Shows the storage path of the STEP 7/PCS 7 project.
- "Instance number (virtual controller)"
  Shows the instance number of the virtual controller.
  The instance number is assigned automatically by SIMIT.

Diagnostics

The virtual controllers are checked when a project is imported.

Errors such as incorrect IP addresses are highlighted in color and reported in plain text in the diagnostics view.

The displays in the diagnostics view disappear automatically when the errors are dealt with. If they do not, click "Refresh" in the distribution editor.

See also

Configuring an additional PC (Page 149)

2.6.2.6 Restarting the virtual controller

As of SIMIT V10.2, a running simulation no longer necessarily needs to be ended when individual VCs lose the connection to SIMIT or need to be restarted for other reasons.

Requirement

- The simulation is running.
- The distribution editor of the virtual controller coupling is open.

Procedure

- Select "Restart" in the shortcut menu of the desired virtual controller.

Result

The virtual controller is restarted with the same configuration as during the initial start.

Note

The "Restart" function is not available in DEMO mode.

2.6.2.7 Configuring an additional PC

Introduction

For complex simulations with multiple virtual controllers, one single PC often does not have sufficient capacity. You can distribute the computing load evenly by integrating additional PCs into the simulation network. You then assign each additional PC one or more virtual controllers in the SIMIT project.

Note

Configuration limits

A SIMIT project can have a maximum of 32 virtual controllers.

One PC can simulate multiple virtual controllers. As a rule of thumb: A CPU core can compute approximately two virtual controllers. The actual CPU power required depends on the complexity of the simulation.

The same version of SIMIT must be installed on all PCs.

Requirement

- The PC is accessible in the network.
- Firewall is set up (Page 50).
- "SIMIT VC" is installed on the PC.
- A coupling of the type "Virtual controller" has been created.
- Virtual controllers from STEP 7/PCS 7 project are imported.
- The "Distribution" editor is open.
- Simulation has stopped.

Procedure

Proceed as follows to configure an additional PC:

1. Change the IP address of the PC "My Computer" from "127.0.0.1" to the IP address at which the PC is accessible in the network.
2. Create an additional PC:
   - Click "New host" in the "Distribution" editor.
   - Select the name of the PC.
3. Drag the desired virtual controller to the PC with drag & drop.
4. In the property view of the virtual controller, go to "Properties > Active addresses" and select the IP addresses that are used in the simulation.
5. Add these IP addresses in the network settings of the PC.
6. In the "Distribution" editor, click on "Save and distribute".

Result

The configuration is saved. The simulation project is now distributed across the PCs.

Conflicts such as duplicate IP addresses and inaccessible PCs are displayed in the property view under "Diagnostics".

See also

Configuring virtual controllers (Page 139)

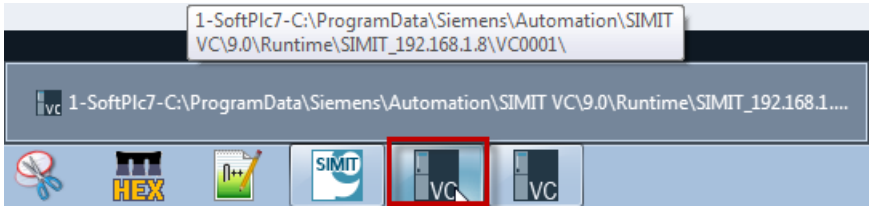
Distribution editor (Virtual Controller) (Page 144)

2.6.2.8 Virtual controller during simulation

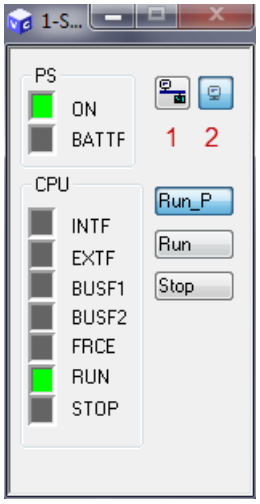
Virtual controller display

During simulation, each virtual controller is displayed as a symbol in the task bar.

The virtual controller symbol displays a tooltip with the path in which the configuration data of the virtual controller is saved.



If you click on a virtual controller symbol, the following display window opens.



The window size is fixed. It must not be closed normally.

To minimize the window, click on the symbol in the task bar again.

The green box to the left of "RUN" indicates that the virtual controller is in RUN.

A red box to the left of "STOP" indicates that the virtual controller is not running.

Virtual controller buttons

- The button above 1 puts the virtual controller online.
- The button above 2 puts the virtual controller offline.
- The Run_P button switches the virtual controller to Run Protect mode. Run-Protect mode of the virtual controller implements the behavior of Run mode.(*)
- The Run button switches the virtual controller to Run mode.
- You switch the virtual controller to Stop mode with the Stop button.(*)

When simulation is ended, all parts of the simulation project are merged on the computer on which SIMIT is installed.

Note

On *

The functions are not available for PA-CPUs. For this reason, the "Run_P" and "Stop" buttons cannot be selected in this case.

See also

"Virtual controller import" dialog box (Page 985)

Properties of the Virtual Controller coupling (Page 152)

Mapping of SIMATIC data types in SIMIT (Page 215)

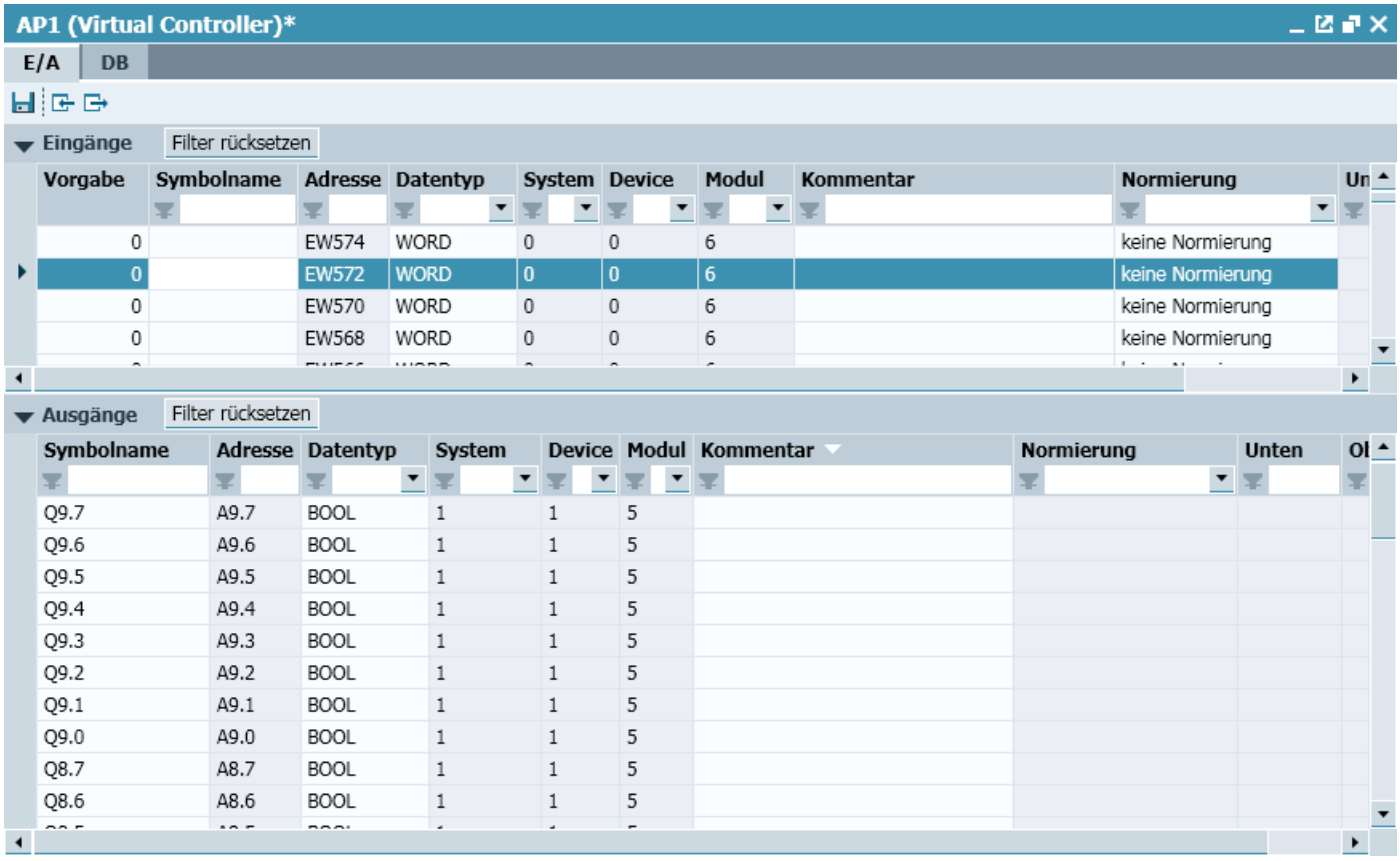
2.6.3 Editing signals

2.6.3.1 Properties of the Virtual Controller coupling

Properties of the address ranges of a virtual controller

The address ranges of the inputs and outputs are taken from the STEP 7/PCS 7 project during import.

"I/O" tab

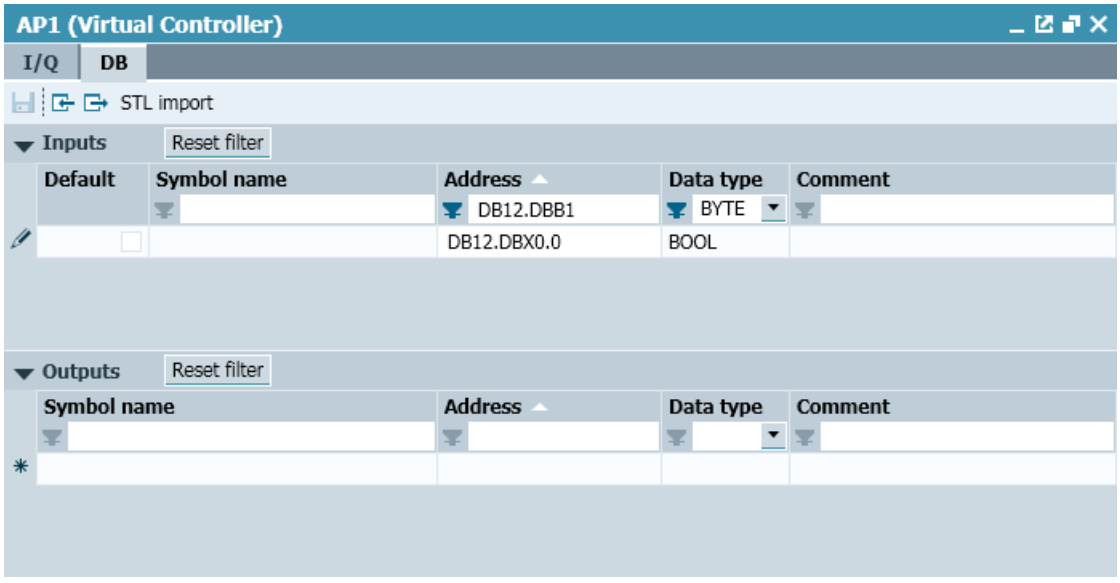


Here the inputs and outputs of the coupling are listed. You can find additional information on this in section: Coupling editor (Page 59).

You can find additional information on importing signal properties in "Importing signal properties" dialog box (Page 988).

You can find additional information on exporting signal properties in "Exporting signal properties" dialog box (Page 991).

"DB" tab



If you need values of PLC data blocks for the simulation, you can enter the addresses or import them using the "STL import" button. You can find additional information on this in section: Importing DB data from STL sources (Page 155).

Note

The addresses are not validated.

Properties of a virtual controller

The following properties are taken from the STEP 7/PCS 7 project during import or entered manually. The properties are not overwritten if you import again.

- "Time slice"
  Specifies the cycle time for execution of OB1.
  The cycle time of each time slice is configured centrally in the "Project settings". You can set the time slice for all virtual controllers at once in the shortcut menu under "Project navigation > Couplings > Virtual controller". This overwrites individual settings.

  Note

  The time slice with the lowest cycle time always has the highest priority, regardless of the numbering.

- "Mnemonics"
  Specifies whether the German ("E/A") or English ("I/Q") term for the absolute addresses is used.

  Note

  Define the mnemonics immediately after import. If you subsequently change the mnemonics, signals already interconnected in a chart can no longer be assigned to the original address.

- "Project name (SIMATIC)"
  Shows the name of the STEP 7/PCS 7 project.
- "Station name (SIMATIC)"
  Shows the station name.
- "Storage path (SIMATIC)"
  Shows the storage path of the STEP 7/PCS 7 project.
- "Serial number of CPU"
  Specifies the serial number of the CPU. The information is only necessary if the STEP 7/PCS 7 project needs the serial number, for example, for licensing parts of the program.
- "Serial number of memory card"
  Specifies the serial number of the memory card. This entry is optional.
- "Cycle time of OBxx"
  Specifies the cycle time.
  The cycle time is applied from the STEP 7/PCS 7 project. OBs with a cycle time shorter than that of the time slice set are run more than once. To reduce load during simulation, set these cycle times to the configured cycle time of the time slice.

Properties of a module

- "Module name"
  Shows the module name.
- "MLFB"
  Shows the Siemens order number.
- "Module slot"
  Shows the module slot.

- "Input address range"
  Shows the address range of the inputs in bytes.
- "Output address range"
  Shows the address range of the outputs in bytes.

See also

Configuring virtual controllers (Page 139)

2.6.3.2 Importing DB data from STL sources

Introduction

The following information is transferred to SIMIT when you import from an STL data source:

- Comment
- When you import inputs: default input assignment

The following data types are applied:

- STRING (implemented as BYTE ARRAY)
- ARRAY
- STRUCT
- UDT
- BOOL
- BYTE
- WORD
- DWORD
- INT
- DINT
- REAL
- CHAR

Note

Always select all utilized STL data sources when you generate the STL sources in STEP 7. If a data block includes a UDT, for example, also add this structure definition when you generate the STL source.

The following data types are included in addressing but not applied:

- S5TIME
- TIME
- DATE

- TIME_OF_DAY
- DATE_AND_TIME

Requirement

- Virtual controller is open.
- The "DB" tab is enabled.
- STL source with data block information exported from STEP 7 is available.

Procedure

Proceed as follows to import DB data from STL sources:

1. Click "STL import".
   The "STL import" dialog box opens.
2. Select the STL source.
3. Enter the data block number.
4. Select the direction.
5. Enter the start and end bytes.

   Note

   Any existing signals in the specified memory area of "From byte" and "Up to byte" are overwritten.

6. Click "Import".

Result

The specified address range of the data block is imported.

See also

"STL import" dialog box (Page 987)

Properties of the Virtual Controller coupling (Page 152)

2.7 PLCSIM Advanced coupling

2.7.1 Operating principle of the PLCSIM Advanced coupling

Operating principle

SIMIT uses the PLCSIM Advanced coupling to communicate with PLCSIM Advanced over a software interface of PLCSIM Advanced. SIMIT uses the PLCSIM Advanced coupling to cyclically exchange data of the I/O area of the configured S7-1500 stations. SIMIT supports up to 16 stations per PC.

Note

You may have to observe additional restrictions and notes in the PLCSIM Advanced documentation.

The following application must be installed on the SIMIT PC for use of the PLCSIM Advanced coupling: This application must also be installed on every other PC that is to be used as the runtime system for PLCSIM Advanced:

- PLCSIM Advanced in a compatible version, see Compatibility tool (www.siemens.com/kompatool)

For access to the STEP 7 project data, the following application must be installed on the configuration computer for STEP 7 (TIA Portal):

- TIA Portal Openness

SIMIT und STEP 7 (TIA Portal) can also be installed on the same PC.

The PLCSIM Advanced application is started and closed together with the simulation.

See also

Creating a coupling of the "PLCSIM Advanced" type (Page 157)

2.7.2 Creating a coupling of the "PLCSIM Advanced" type

Introduction

In a SIMIT project, you can create a coupling of the type "PLCSIM Advanced". You create "stations" under the coupling to which you import the hardware configuration of a station from STEP 7.

Requirement

- A project is open in SIMIT.
- STEP 7 (TIA Portal) and TIA Portal Openness are installed on the SIMIT PC in a compatible version, see Compatibility tool (www.siemens.com/kompatool), OR
- The HWCNExport file is available.
  Creating an HWCNExport file from TIA projects (Page 64)

Procedure

Follow these steps to create a coupling of the "PLCSIM Advanced" type:

1. Add a new coupling under "Couplings" in the project tree.
   The "Selection" dialog box opens.
2. Select the coupling type "PLCSIM Advanced".
   The "PLCSIM Advanced Import" dialog box opens.
3. Configure the import of stations and devices from a STEP 7 project or the HWCNExport file.

Result

You can find the result after import of a STEP 7 project in Importing hardware configuration to station (Page 158).

If you complete the coupling of the type "PLCSIM Advanced" without importing a STEP 7 project, only the "PLCSIM Advanced" folder is created in the project tree under couplings.

See also

"PLCSIM Advanced import" dialog box (Page 984)

HWCNExport introduction (Page 62)

2.7.3 Importing hardware configuration to station

Introduction

You import the hardware configuration directly from STEP 7. Import the hardware configuration to the station in SIMIT again if you have changed the hardware configuration in STEP 7.

When importing the hardware configuration from the TIA Portal, the following structures are evaluated in addition to the simple data types and imported as WORD:

- PD_ZSW1
- PD_ZSW2
- PD_Gx_ZSW
- PD_STW1
- PD_STW2

- PD_Gx_STW
- PD_MELDW

Requirement

- A "PLCSIM Advanced" type coupling has been created.
- STEP 7 is installed on the SIMIT PC in a compatible version, see Compatibility tool (www.siemens.com/kompatool).
- Hardware configuration is compiled and loaded in STEP 7.

Procedure

Follow these steps to import the hardware configuration to a station:

1. Select the "Import" command from the "Couplings > PLCSIM Advanced" shortcut menu in the project tree.
   The "PLCSIM Advanced import" dialog box will open.
2. Select the source from which you want to import the hardware configuration.
3. Select the mode for importing the symbols.
4. Select the desired stations and devices.
5. Click "Import".

Result

A station is created for each station from the STEP 7 project in the project tree under "Couplings > PLCSIM Advanced". During the initial import the imported stations are assigned to the emulation PC "Own computer" in the "Distribution" editor.

See also

"PLCSIM Advanced import" dialog box (Page 984)

Creating a coupling of the "PLCSIM Advanced" type (Page 157)

HWCNExport introduction (Page 62)

Editing the signals of a coupling (Page 210)

2.7.4 Configuring a PLCSIM Advanced coupling station

Requirement

- Station has been created.
- Hardware configuration has been imported.

Procedure

Follow these steps to configure a station of a PLCSIM Advanced coupling:

1. Open the station.
2. In the properties of the station, select one of the following options from the "Time slice" drop-down list:
   - Time slice 1 to 8.
   - Bus synchronous
   - Event synchronous

   The options "Bus synchronous" and "Event synchronous" are only displayed if they are available in the hardware configuration of the PLCSIM Advanced.
   If "Bus synchronous" or "Event synchronous" has been selected, additional properties for the configuration are displayed.
   If a synchronous event is activated and no suitable time slice is configured in the project manager, an alarm is displayed.
3. Open the distribution editor of the PLCSIM Advanced coupling.
4. Configure each station in the properties window:
   - Enter a name if required.
   - Specify the communication path.

Result

The station is configured. Import (Page 227) signal properties to add input and output signals. You can edit signal properties.

You can find additional information on editing and exporting signals in:

- Editing the signals of a coupling (Page 210)
- Exporting signal properties (Page 228)

See also

Distribution editor (PLCSIM Advanced) (Page 160)

Importing hardware configuration to station (Page 158)

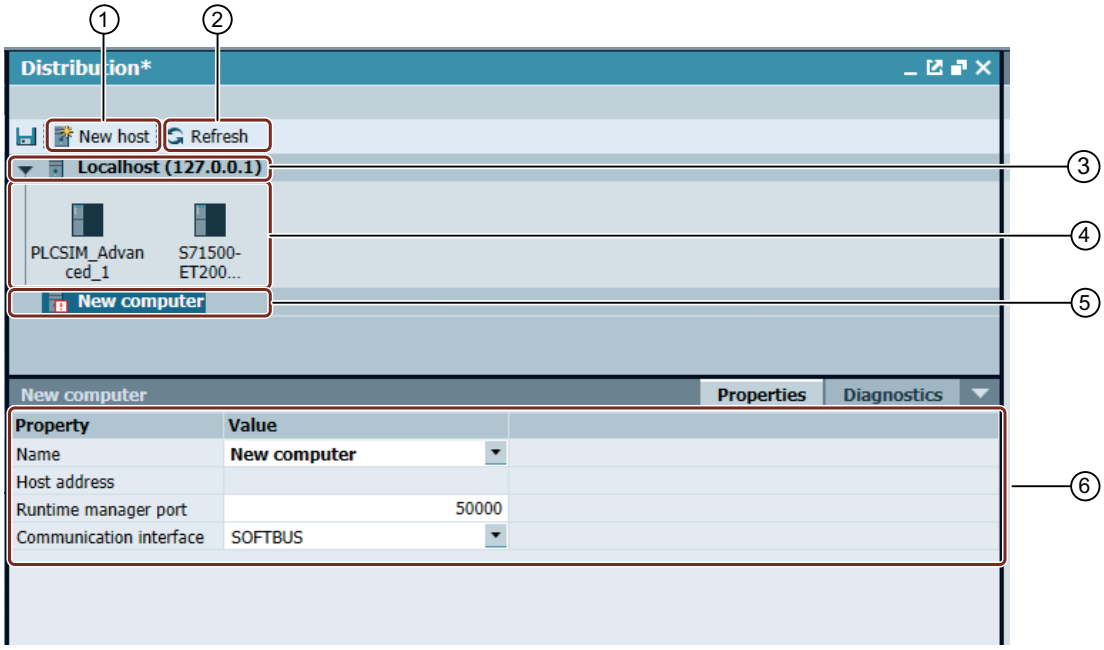
2.7.5 Distribution editor (PLCSIM Advanced)

Distribution editor

In the distribution editor, you configure the stations of a PLCSIM Advanced coupling and distribute them to additional PCs as needed.

This spreads the computing load for simulation across multiple PCs.

Double-click the distribution editor to open it in the project navigation under "Couplings > PLCSIM Advanced".



① Creation of a new host
② Updates the list of computers
③ Represents the PC on which the SIMIT project is open.
④ Stations that you have imported from a project. The stations are assigned to "My Computer" by default.
⑤ Additional PC (optional) with drop-down list of computers on which a Runtime Manager is installed.
⑥ Property view

Menu bar

You can execute the following functions in the menu bar of the distribution editor:

- "Save"
  Saves the configuration.
- New host
  Additional PCs are added.
  From the list, you can select available PCs that have already installed a "Runtime Manager".
  If a PC is added on which the "Runtime Manager" of PLCSIM Advanced is not yet installed, this needs to be installed.
  You distribute the available PLCSIM Advanced-instances among the PCs with drag-and-drop.
- Refresh

Properties of computers

- Name
  Specifies the name of the PC.
  It is also possible here to select from the drop-down list an available PC that has already installed a "Runtime Manager".
- Host address
  Shows the IP address of the PC by which it is currently identified in the network.
  The additional PC must be located in the same network as "My Computer".
- Runtime manager port
  Specifies the port of the Runtime Manager on an additional PC.
- Communication interface
  Specifies the communication path between PLCSIM Advanced and SIMIT. You can find additional information on the communication paths in the PLCSIM Advanced documentation.

Properties of a station

- Name
  Specifies the station name in the SIMIT project. Change the name in the project navigation using the station shortcut menu.
- Simatic project name
  Shows the name of the project.
- Simatic station name
  Shows the station name.
- Simatic project path
  Shows the storage path of the project.
- Instance number (VC)
  Shows the instance number.
  The instance number is assigned automatically by SIMIT.
- Managed by SIMIT
  Specifies whether the instance name of SIMIT is assigned automatically.
- Instance name
  Shows the instance name.
  The instance name is either assigned automatically by SIMIT or assigned by the user.

Shortcut menu of a station

- Clear memory
  Deletes the project that has been loaded to the station.

See also

Configuring a PLCSIM Advanced coupling station (Page 159)

2.8 PLCSIM coupling

2.8.1 How the PLCSIM coupling works

SIMIT uses the PLCSIM coupling to communicate with PLCSIM over a PLCSIM software interface (Prosim). When the PLCSIM coupling is used, PLCSIM and SIMIT must be installed on the same computer. PLCSIM must be started before you start a simulation in SIMIT with the PLCSIM coupling. You must not end PLCSIM while the simulation is running, as the connection will otherwise be canceled and may not be re-established.

Note

To use the PLCSIM coupling, you also need the PLCSIM software version 5.4 SP5 or higher. This software is not included in the SIMIT product package.

PLCSIM is a program that can be used independently of SIMIT and can be loaded from SIMATIC Manager with a STEP 7 program. PLCSIM will then simulate a SIMATIC controller. In order to simulate the controller behavior as realistically as possible, we recommend that you always load the STEP 7 program along with the system data blocks (SDBs) into PLCSIM.

Note

The hardware configuration is also determined through the system data. In the SIMIT coupling with PLCSIM, data exchange is only possible for I/O signals that are defined by the configured hardware.

2.8.2 Configuring the PLCSIM coupling

2.8.2.1 Creating a PLCSIM coupling

To create a PLCSIM coupling, select the option "PLCSIM" in the "Selection" dialog box.

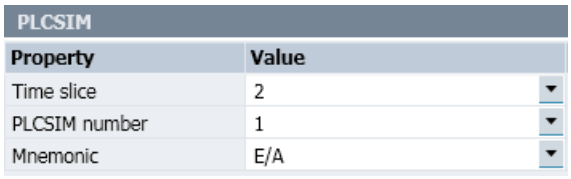
2.8.2.2 Configuring I/O signals in the PLCSIM coupling

The options for entering input/output signals for the coupling are as follows:

- You enter the signals manually in the coupling editor.
- You import the symbol table from your SIMATIC project. You can find additional information on this in section: "Importing signal properties" dialog box (Page 988).

2.8.2.3 Properties of the PLCSIM coupling

Once the coupling is opened, the coupling editor appears in the work area. You can define the following properties in the property view:



PLCSIM	
Property	**Value**
Time slice	2
PLCSIM number	1
Mnemonic	E/A

- **Time slice**
  The cycle in which the coupling exchanges data is set here. The assignment of absolute cycle times to the 8 possible time slices applies to the entire project. Time slice 2, which corresponds to a cycle of 100 ms, is preset.

  Note

  The time slice with the lowest cycle time always has the highest priority, regardless of the numbering.

- **PLCSIM number**
  You can create up to 8 PLCSIM couplings in a project to connect to the corresponding instances of PLCSIM. Select the number of the PLCSIM instance with which the coupling is to communicate here. The number can be found in the PLCSIM window header line.
  If you set the PLCSIM number to "Not assigned", no connection with PLCSIM will be established when the simulation is started and there will be no data exchange.
- **Mnemonics**
  Here, you select whether the international (I/Q) or German (E/A) mnemonics are to be used for the inputs and outputs.

Additional functions

Copying and pasting scaling

You can find additional information in section: Transferring scaling to another signal (Page 239).

Importing signal properties

You can find additional information in section: "Importing signal properties" dialog box (Page 988).

Exporting signal properties

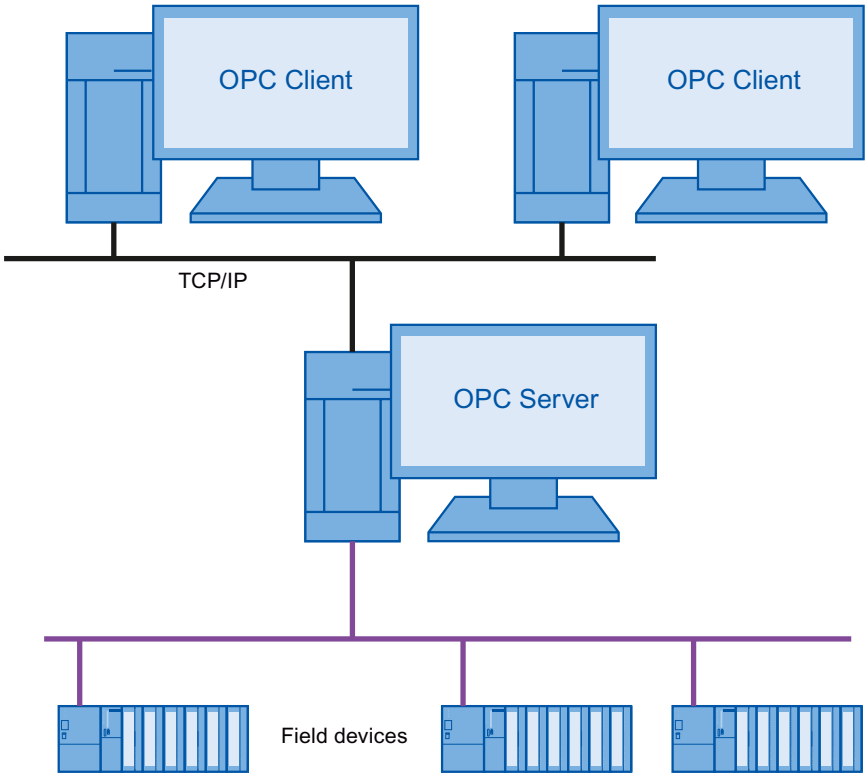
You can find additional information in section: "Exporting signal properties" dialog box (Page 991).

2.9 OPC coupling

2.9.1 How the OPC coupling works

Introduction

OPC is a widely used communication standard in the automation sector, and is maintained and supported by the OPC Foundation (www.opcfoundation.org). The OPC protocol is used for device-independent data exchange between programs with OPC capability.



In the OPC Classic Architecture, an OPC server is connected to a subordinate device layer whose signals are made available via the local network to one or more OPC clients.

Note

To protect the confidentiality of your data, use the OPC coupling only as part of a holistic, state-of-the-art industrial security concept. This should take into account products from other manufacturers that are being used.

You can find additional information on the Internet (https://www.siemens.com/industrialsecurity).

OPC protocols supported

The SIMIT OPC coupling supports the following OPC protocols:

- OPC DA 3.0 in "Data Access" mode
- OPC UA Specification 1.04, the function is limited to the "Data Access" (DA) information model.

OPC couplings in SIMIT

SIMIT supports the following OPC coupling:

OPC coupling	Number per project
OPC DA server coupling	1
OPC DA client coupling	Up to 32[1]
OPC UA server coupling	
OPC UA client coupling	

[1] For different OPC servers or different name spaces of OPC UA servers

The OPC DA server and OPC DA client can be operated locally on the same PC or on different PCs. An OPC DA server and various OPC client couplings can also be created simultaneously in a SIMIT project.

Browsing functionality

SIMIT supports the browsing functionality which allows you to apply the data provided by an OPC server automatically to the signal table.

See also

Principle of operation of the SIMIT OPC DA server (Page 171)

Principle of operation of the OPC UA client coupling (Page 181)

2.9.2 Signal transfer between OPC server and OPC client

Establishment of connection between OPC server and OPC client

When you launch a simulation, connections are established to all configured OPC servers. If no connection can be established after 5 seconds, the simulation will start anyway. The connection attempts continue to run in the background. Status information is displayed over the SIMIT message system. You can also query the status of each OPC coupling with the "Status display" signal.

Start of signal transfer

Once a connection is established, the signals are registered at the OPC server for each OPC client. The cause of unregistered signals is frequently one of the following:

- Wrong data type
- Signal unknown at the OPC server.

Following registration, the signals are transferred cyclically between OPC server and OPC client. The following factors are key to updating values:

- OPC server cycle time
- OPC client coupling time slice
- Cycle multiplier of a signal

Cycle multiplier

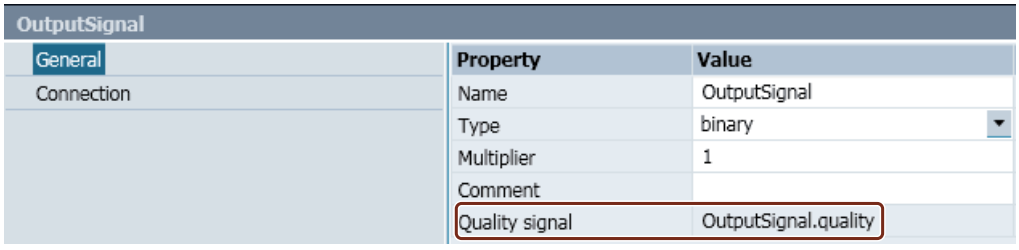
The cycle multiplier is an integer, "n", which only updates values in every nth cycle. By specifying a cycle multiplier, you the reduce the load on the communication connection between OPC server and OPC client. You can configure the cycle multiplier separately for each signal provided by the OPC server.

- Signal changes frequently: Small cycle multiplier
- Signal changes slowly: Large cycle multiplier

Example: For an OPC client coupling, a cycle time of 250 ms is set with the time slice. A signal is configured with a cycle multiplier of "4". The signal is therefore only updated every 250 ms × 4 = 1000 ms.

Quality code

Signals transferred between OPC server and OPC client by default also return a "quality code" that indicates the quality of the signal. For each output signal of an OPC client coupling, SIMIT by default creates an integer "quality signal" containing the quality code. The signal name is structured as follows: <Name of output signal>.<quality>



You can use the quality signal in charts, like all other signals. While a simulation is running, the "quality code" is displayed in the properties of each quality signal.

Signals transferred from the SIMIT OPC client to an OPC server output the following quality code:

- "Good, non-specific". This corresponds to "0xC0" or "192".

Signals transferred from the SIMIT OPC UA server to OPC UA client return the following quality code:

- "Good". The numerical counterpart is "0x00" or "0".

2.9.3 Mapping of signals and data types of an OPC server

Mapping of data types

SIMIT only has binary, integer and analog signals, with integer and analog signals having a data width of 8 bytes.

An OPC server can provide signals in a range of other data types. The table below shows the assignment of OPC data types to the data types used in SIMIT.

OPC DA server (General)	**OPC UA server (General)**	**OPC UA server (SIMIT)**	**SIMIT**
BOOL, UI1	Boolean	Boolean	binary
I1, I2, I4, I8, UI1, UI2, UI4, UI8	SByte, Int16, Int32, Int64, Byte, UInt 16, UInt32, UInt64	Int32, Int64	integer
R4, R8	Float, Double	Double	analog

There is no data loss in conversion for output signals of the coupling, as integer and analog signals in SIMIT have the maximum data width (8 bytes).

Mapping of signals

The signals of the OPC server can be read or written or both. They are treated as inputs or outputs in SIMIT depending on this read/write access.

OPC	**OPC UA server (SIMIT)**	**SIMIT**
readable	CurrentRead / Current-Write	Output
writeable	CurrentRead	Input
readWriteable	CurrentRead	Input

2.9.4 Status of OPC servers

Core statement

While simulation is running, the properties window of the OPC client coupling displays the current status of the OPC server addressed. The status is also output with an output signal that you define when you configure the coupling.

Note

Which of these status values an OPC server provides depends on the OPC server implementation.

OPC DA server status

The following table shows the numerical values and their meaning for the OPC DA implementation.

Status value [1]	Meaning [2]
0	No connection to OPC server
1	The server is running normally. This is the usual state for a server.
2	A vendor specific fatal error has occurred within the server. The server is no longer functioning. The recovery procedure from this situation is vendor specific. An error code of E_FAIL should generally be returned from any other server method.
3	The server is running but has no configuration information loaded and thus cannot function normally. Note this state implies that the server needs configuration information in order to function. Servers which do not require configuration information should not return this state.
4	The server has been temporarily suspended via some vendor specific method and is not getting or sending data. Note that Quality will be returned as OPC_QUALITY_OUT_OF_SERVICE.
5	The server is in Test Mode. The outputs are disconnected from the real hardware but the server will otherwise behave normally. Inputs may be real or may be simulated depending on the vendor implementation. Quality will generally be returned normally.
6	The server is running properly but is having difficulty accessing data from its data sources. This may be due to communication problems, or some other problem preventing the underlying device, control system, etc. from returning valid data. It may be complete failure, meaning that no data is available, or a partial failure, meaning that some data is still available. It is expected that items affected by the fault will individually return with a BAD quality indication for the items.

[1] The information on status values 1 to 6 is taken from the "Data Access Custom Interface Standard, Version 3.00" specifications.

[2] The description of status values 1 to 6 is taken from the OPC specification.

OPC UA server status

The following table shows the numerical values and their meaning for the OPC UA implementation.

Status value	ID	Meaning [1]
0	Running	The server is running normally. This is the usual state for a server.
1	Failed	A vendor-specific fatal error has occurred within the server. The server is no longer functioning. The recovery procedure from this situation is vendor specific. Most Service requests should be expected to fail.
2	NoConfiguration	The server is running but has no configuration information loaded and therefore does not transfer data.
3	Suspended	The server has been temporarily suspended by some vendor-specific method and is not receiving or sending data.
4	Shutdown	The server has shut down or is in the process of shutting down. Depending on the implementation, this might or might not be visible to clients.
5	Test	The server is in Test Mode. The outputs are disconnected from the real hardware, but the server will otherwise behave normally. Inputs may be real or may be simulated depending on the vendor implementation. **StatusCode** will generally be returned normally.
6	CommunicationFault	The server is running properly, but is having difficulty accessing data from its data sources. This may be due to communication problems or some other problem preventing the underlying device, control system, etc. from returning valid data. It may be a complete failure, meaning that no data is available, or a partial failure, meaning that some data is still available. It is expected that items affected by the fault will individually return with a BAD FAILURE status code indication for the items.
7	Unknown	The server is running properly, but is having difficulty accessing data from its data sources. This may be due to communication problems or some other problem preventing the underlying device, control system, etc. from returning valid data. It may be a complete failure, meaning that no data is available, or a partial failure, meaning that some data is still available. It is expected that items affected by the fault will individually return with a BAD FAILURE status code indication for the items.

[1] The description of status values 1 to 7 is taken from the description of the utilized SDK (Unified Automation .NET based OPC UA Client & Server SDK Bundle V2.6.1).

See also

Properties of the OPC DA client coupling (Page 175)

2.9.5 Configuring an OPC DA client/server coupling

2.9.5.1 Principle of operation of the SIMIT OPC DA server

Behavior of the SIMIT OPC DA server

The SIMIT OPC DA server is automatically configured with the signals configured in the coupling when you launch the simulation. From this point, OPC clients can connect to the server and access the signals.

When you end simulation, the SIMIT OPC DA server remains active until all OPC DA clients have logged off. Signals are no longer updated in this state.

If you change the configuration of the SIMIT OPC DA server, the configuration will automatically be transferred the next time simulation is launched. Requirements: The SIMIT OPC DA server is not connected to an OPC DA client.

Quality codes of output OPC signals

The SIMIT OPC DA server transfers OPC signals with the following quality codes:

Quality code	Requirement
"Good, non-specific"; 0xC0	Simulation launched
"Bad, Out of service"; 0x1C	Simulation ended, but at least one OPC DA client still connected. The SIMIT OPC DA server is not disabled until all OPC DA clients are disconnected.

Data transmission

The SIMIT OPC DA server checks regularly in the simulation model whether the values have changed. The SIMIT OPC DA server supports synchronous and asynchronous data transmission:

- Synchronous data transmission All values are cyclically transmitted.
- Asynchronous data transmission Only changed values are transmitted. The SIMIT OPC DA server automatically groups values. Criteria for grouping are:
  - Data type and multiplier, each for input and output values

  All values in a group are generally transmitted in case a value has changed.

Status messages

The SIMIT OPC DA server outputs the following status that can be queried by an OPC DA client:

Value	Meaning	Requirement
1	Running	Simulation launched
4	Suspended	Simulation ended

Data types provided

The following data types are provided in the SIMIT OPC DA server:

SIMIT	OPC
Binary	BOOL
Integer	I8 if the option "Use 64-bit integer" is selected, otherwise I4
Analog	R8

See also

How the OPC coupling works (Page 165)

2.9.5.2 Configuring a SIMIT OPC DA server coupling

Requirements

- Project is open in SIMIT.

Procedure

Follow these steps to create a SIMIT OPC DA server coupling:

1. Select the "New coupling" command from the "Couplings" shortcut menu in the project navigation.
2. Select the coupling type "OPC DA server".
3. Select a time slice with an update cycle of at least 100 ms.
4. If the OPC DA client and OPC DA server are installed on different PCs, please see the "Notes on DCOM configuration (Page 176)".

Result

The SIMIT OPC DA server is configured. The following options are available for adding input and output signals:

- Entering signals manually
- Importing the signal table (Page 227)

See also

Properties of the SIMIT OPC DA server coupling (Page 174)

Mapping of signals and data types of an OPC server (Page 168)

2.9.5.3 Configuring an OPC DA client coupling

Introduction

The OPC DA client coupling only supports asynchronous data transmission from one OPC DA server to another. An OPC DA server automatically groups values. All values in a group are transmitted in case a value changes.

Requirements

- Project is open in SIMIT.
- ProgID of the OPC DA server is known.
- Host name of the PC on which the OPC DA server is installed is known.
- OPC DA server can be accessed in the network.

Procedure

Follow these steps to create an OPC DA client coupling:

1. Select the "New coupling" command from the "Couplings" shortcut menu in the project navigation.
2. Select the coupling type "OPC DA client".
3. Configure the connection to the OPC DA server.
4. If the OPC DA client and OPC DA server are installed on different PCs, please see the "Notes on DCOM configuration (Page 176)".

Result

The coupling with the OPC DA server is configured. The following options are available for adding input and output signals:

- Reading signals from an OPC server (Page 241)
- Importing the signal table (Page 227)
- Entering signals manually

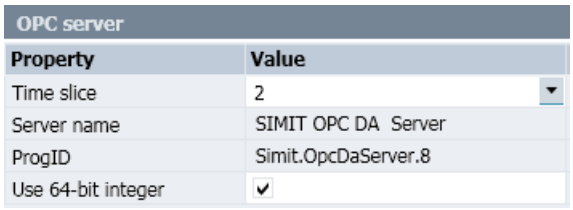
See also

Properties of the OPC DA client coupling (Page 175)

Editing the signals of a coupling (Page 210)

2.9.5.4 Properties of the SIMIT OPC DA server coupling

Once the coupling is opened, the coupling editor appears in the work area. You can define the following properties in the property view:



- **Time slice**
  This is where you set the cycle in which the coupling exchanges data. The assignment of absolute cycle times to the 8 possible time slices applies to the entire project. The default is time slice 2, which corresponds to a cycle of 100 ms in the standard setting.

  Note

  The time slice with the lowest cycle time always has the highest priority, regardless of the numbering.

  The OPC server updates its data every 100 ms. Cycle times below this value should therefore not be set.

- **Server name**
  The SIMIT OPC server name is displayed for information only. It is set to "SIMIT OPC DA Server". The server name or ProgID is needed to configure an OPC client for access to this OPC server.
- **ProgID**
  The ProgID of the SIMIT OPC server is displayed for information only. It is set to "Simit.OpcDaServer.8". The ProgID or server name is needed to configure an OPC client for access to this OPC server.
- **Use 64-bit integers**
  Integer values in SIMIT have a data width of 64 bits (8 bytes). They are therefore also saved in this format in the OPC server coupling.
  Deselect this option if an OPC client connected to this OPC server cannot process integer values with a data width of 8 bytes. SIMIT then transfers all integer values with a data width of 32 bits (4 bytes) only. Note that this may lead to loss of data if a number cannot be represented with 32 bits.

See also

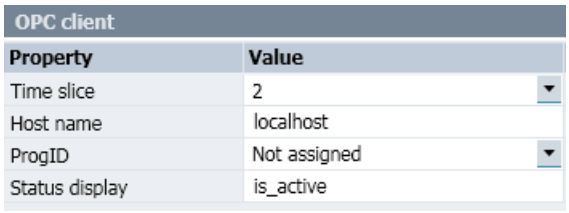
"Importing signal properties" dialog box (Page 988)

"Exporting signal properties" dialog box (Page 991)

Configuring a SIMIT OPC DA server coupling (Page 172)

2.9.5.5 Properties of the OPC DA client coupling

Once the coupling is opened, the coupling editor appears in the work area. You can define the following properties in the property view:



- **Time slice**
  This is where you set the cycle in which the coupling exchanges data. The assignment of absolute cycle times to the 8 possible time slices applies to the entire project. The default is time slice 2, which corresponds to a cycle of 100 ms in the standard setting.

  Note

  The time slice with the lowest cycle time always has the highest priority, regardless of the numbering.

- **Host name**
  The name of the computer on which the OPC server is running. This is the OPC server with which the coupling communicates. You can provide either the computer name or its IP address. The default is the local computer (localhost).
- **ProgID**
  Here, enter the ProgID of the OPC server with which the OPC client is to connect on the computer specified under host name.
  If the connection is local, i.e. "localhost" is entered under host name, the ProgIDs of the OPC servers that are available locally will be available for selection. Select the OPC server with which the coupling is to communicate.
  If the ProgID is set to "Not assigned", no connection to an OPC server is established when the simulation is started. The coupling signals are not updated cyclically.
- **Status display**
  Specifies the name of the integer output signal that provides the status of the connected OPC server while simulation is running. You can use this output signals to evaluate the server status in your SIMIT project. One output signal is generated for each OPC DA client coupling.

See also

"Importing signal properties" dialog box (Page 988)

"Exporting signal properties" dialog box (Page 991)

Status of OPC servers (Page 169)

Configuring an OPC DA client coupling (Page 173)

2.9.5.6 Notes on DCOM configuration

Basics of DCOM configuration

No further adjustments to the operating system configuration are required if you have installed the OPC DA server and OPC DA client on **the same** PC.

If you have installed the OPC DA server and OPC DA client on **different** PCs, you will need to check and if necessary adjust the following settings in the operating system:

- Firewall configuration
- Domains and users
- Assignment of rights

Note

All settings in the rights systems of a Windows PC depend greatly on the operating system version and the programs already installed on your PC. We can therefore only give you a guide and not precise instructions for the configuration of DCOM in your specific environment.

Note

Only change settings on your PC if you know exactly what they mean. Make sure that you do no inadvertently leave your PC unprotected.

Firewall configuration

If you use a firewall, you will need to enable TCP port 135 for incoming queries for DCOM. You must ensure that none of the OPC servers or OPC clients or the two programs "Microsoft Management Console" and "OPCEnum" are blocked.

Domains and users

All relevant PCs must belong to the same working group or domain. Use the same user name and the same password on all PCs.

Assignment of rights

When an OPC connection is established over DCOM, a PC accesses the resources of or launches specific processes on another PC. You will need specifically to allow this, as it involves processes that could potentially put your PC at risk.

Both the access rights and start and activation rights need to be correctly set. Recommendation: Set up a new user group with these advanced access rights. Assign the users who use the OPC connection to this user group.

As well as making general COM security settings, you also need to set the rights of the OPC server and the system utility program "OPCEnum.exe".

The program "dcomcnfg.exe" under Windows 7 is used for configuration.

Details of DCOM configuration can be found in "Using OPC via DCOM with Windows XP Service Pack 2" from the OPC Foundation, available on the Internet at http://www.opcfoundation.org.

See also

Configuring a SIMIT OPC DA server coupling (Page 172)

Configuring an OPC DA client coupling (Page 173)

2.9.6 Configuring the OPC UA client/server coupling

2.9.6.1 Operating principle of the SIMIT OPC UA server

Introduction

You can assign up to 32 OPC UA server couplings in SIMIT.

The server functions defined in the OPC UA standard are provided by SIMIT as a Windows Service.

The connection to SIMIT is established via the configuration of the corresponding I/O coupling.

The SIMIT OPC UA server supports secured data transmission.

Note

During setup, there is limited visibility of the access points to the local computer provided by the service (127.0.0.1).

Secured data transmission

During secured data transmission, the OPC UA server and OPC UA client are authenticated by exchanging certificates. The OPC UA server certificate is automatically generated during installation of SIMIT.

The OPC UA client must also provide a valid certificate.

By default, only users of the "Administrators" user group are authorized to manage certificates on the SIMIT PC.

Only users belonging to this group have the necessary authorization to access the certificate store.

User authentication is not supported by the SIMIT OPC UA server. The OPC UA client must therefore use the user name "**Anonymous**" for logon.

The SIMIT OPC UA server supports the following security policies for secured data transmission.

- Security Policies:
  - None, Basic256, Basic256Sha256
- Message Security Modes:
  - None, Sign, Sign&Encrypt

Behavior of the OPC UA server

As mentioned in the introduction, the actual server functionality runs in the context of a Windows service. Each configured SIMIT OPC UA server coupling establishes control and data connections to this service at runtime via IPC mechanisms (Inter Process Communication).

When you start the simulation, the SIMIT OPC UA server will automatically be configured with the signals that are configured in the coupling.

All signals belonging to a coupling are managed in a unique namespace assigned to the coupling.

The address (URI) of this namespace is a property of the OPC UA server coupling and can be queried by OPC UA clients or be used directly in the OPC UA client.

During runtime of the simulation, OPC UA clients can query the namespaces managed by the OPC UA server and access the signals available there.

The access can be read-only ("polling"), or the OPC UA client can register with the server for notification of changes to specific signals ("subscription").

If you exit the simulation, the SIMIT OPC UA server remains active, but the namespaces of the utilized couplings are no longer available.

Signals are no longer updated in this state.

If you change the configuration of the SIMIT OPC UA server coupling, the new configuration is automatically transferred the next time a simulation is started.

Quality codes of supplied OPC UA signals

The SIMIT OPC UA server transfers the signals with the following quality codes:

Quality code	Requirement
"BadWaitingForInitialData"; 0x80320000	Simulation started, waiting for valid data
"Good";0x00	Simulation started
"BadNoData"; 0x809B0000	Simulation ended
"UncertainLastUsableValue"; 0x40900000	Communication temporarily disrupted, last valid value

Data transmission

The SIMIT OPC UA server directly accesses the signals provided cyclically by the SIMIT OPC UA server coupling. The signals provided to the SIMIT simulation model are updated simultaneously.

This happens for both signal directions in each cycle in the time slice assigned to the coupling.

The utilized OPC UA framework allows the OPC UA-compliant exchange of this "real life" data with the SIMIT simulation model.

Status messages

The SIMIT OPC UA server supplies the following statuses, which can be queried by an OPC UA client:

Value	Meaning	Requirement
0	Running	Simulation started

Provided data types

The following data types are provided in the SIMIT OPC UA server:

SIMIT	OPC UA
Binary	Boolean
Integer	Int64, if option "Use 64-bit integer" is selected, otherwise Int32
Analog	Double

2.9.6.2 Setting up the firewall for OPC UA server

Requirement

Due to security policies, the firewall settings for the SIMIT OPC UA server are limited to local access during the setup.

- SIMIT is installed.
- All nodes are located within the same network.

Note

Read the security notes in the preface.

Changes to the firewall may affect the security of your system!

If necessary, contact your system administrator before performing the steps described in the following.

Procedure

1. Open the "Windows Defender Firewall with Advanced Security", for example, by entering "wf.msc" in the system prompt and confirming with "Enter".
2. Click on "Inbound Rules".
3. Double-click "OPC UA Server Service".
   The "Properties of OPC UA Server Service" window opens.
4. Select the "Scope" tab.

5. Select "Remote IP address" > "Any IP address". Alternatively, add the IP addresses of the systems involved to the list of allowed remote IP addresses.
6. Confirm with "OK".

2.9.6.3 Configuring a SIMIT OPC UA server coupling

Requirement

- Project is open in SIMIT.
- The needed permissions for accessing the certificate store conform to the policies described further below.
  Certificates (Page 189)

Procedure

To create a SIMIT OPC UA server coupling, follow these steps:

1. Select the "New coupling" command from the "Couplings" shortcut menu in the project tree.
2. Select the coupling type "OPC UA server".
3. If you want to allow secured data transmission:
   - If the certificate of the OPC UA client is available, copy it to the folder "..\trusted\certs"
   - If the client was refused when establishing the connection, copy the certificate sent by the OPC UA server from the "..\rejected\certs" folder to the "..\trusted\certs" folder.

Note

Please note: Unless permissions are changed accordingly, these actions can only be performed by a user with Administrator authorization.

Result

The SIMIT OPC UA server coupling is configured. The following options are available for adding input and output signals:

- Entering signals manually
- Importing a signal table

2.9.6.4 Properties of the OPC UA server coupling

Once the coupling is opened, the coupling editor appears in the work area. You can define the following properties in the property view:

- **Time slice**
  Specifies the time interval at which the coupling cyclically exchanges data. The assignment of absolute cycle times to the 8 available time slices is valid for the entire project. Time slice 2 is preset, which corresponds to a 100 ms cycle in the default setting.

  Note

  The time slice with the lowest cycle time always has the highest priority, regardless of the numbering.

- **OPC UA server URL**
  Specifies the address at which the OPC UA server can be reached. This address is assigned at the time of installation and cannot be changed. The URL consists of the following:
  - Log
  - Server name
  - Port
- **Namespace URI**
  Specifies the basis of the namespace All signals of this coupling are published under the namespace, which results from this URI with the coupling name added. This namespace cannot be changed.
  In the example presented above:
  urn:md15qxjc:Siemens:Simit.OpcUaServer.OPC UA Server
  This namespace cannot be changed. The uniqueness of the signals is guaranteed by including the coupling name as a component of the namespace.
- **Use 64-bit integer**
  Integer values in SIMIT have a data width of 64 bits (8 bytes). They are therefore also created in the OPC UA server coupling in this format.
  If an OPCUA client that is connected to this OPC UA server cannot handle integer values with a data width of 8 bytes, deselect this option. SIMIT then transmits all integer values with a data width of only 32 bits (4 bytes). Note that this may lead to loss of data if a number cannot be represented with 32 bits.
- **Status display**
  Specifies the name of the integer output signal in which the status of the connected OPC UA server is provided during an active simulation. You may use this output signal for evaluating the server status in your SIMIT project. One output signal each is generated for each OPC UA server coupling.

2.9.6.5 Principle of operation of the OPC UA client coupling

Introduction

You can create up to 32 OPC UA client couplings in SIMIT. Each OPC UA client can contact an OPC UA server and exchange data.

The OPC UA client coupling supports secure data transfer.

Secure data transfer

In secure data transfer, the OPC UA server and OPC UA client authenticate themselves by exchanging certificates. The OPC UA client certificate is generated automatically during SIMIT installation. On the SIMIT PC, only the user group "Administrators" is authorized by default to configure an OPC UA client coupling with secure data transfer.

See also

Mapping of signals and data types of an OPC server (Page 168)

How the OPC coupling works (Page 165)

Certificates (Page 189)

2.9.6.6 Configuring an OPC UA client coupling

Requirements

- Project is open in SIMIT.
- URL of the PC with installed OPC UA server is known.
- The OPC UA server is accessible in the network.
- The OPC UA server supports either the authentication method "Anonymous" or "User/Password".
- Access rights to the following directories are available:
  - "C:\ProgramData\Siemens\Automation\SIMIT\8.0\PKI\own\private": Read
  - "C:\ProgramData\Siemens\Automation\SIMIT\8.0\PKI\trusted\certs": Write

Procedure

To create an OPC UA coupling, follow these steps:

1. Select the "New coupling" command from the "Couplings" shortcut menu in the project navigation.
2. Activate the coupling type "OPC UA client".
3. Configure the connection to the OPC UA server.
4. If you configure secure data transfer:
   - Copy the certificate sent by the OPC UA server from the "..\rejected\certs" folder to the "..\trusted\certs" folder.
   - On the OPC UA server, add the OPC UA client to the "Trusted clients" list.

Note

Please note: Unless permissions are changed accordingly, these actions can only be performed by a user with System Administrator authorization.

Result

The coupling with the OPC UA server is configured. The following options are available for adding input and output signals:

- Reading signals from an OPC server (Page 241)
- Importing a signal table (Page 227)
- Entering signals manually

See also

Properties of the OPC UA client coupling (Page 184)

Mapping of signals and data types of an OPC server (Page 168)

Error messages upon connection to the OPC UA server (Page 183)

Certificates (Page 189)

Editing the signals of a coupling (Page 210)

2.9.6.7 Error messages upon connection to the OPC UA server

OPC UA client: Error messages and remedies

Error message	Cause	Remedy
"The client application does not have a certificate assigned. Secured connections are not possible."	You may not have read access to the following directory: • "C:\ProgramData\Siemens\Automation\SIMIT\8.0 \PKI\own\private"	Have the administrator set up read access to the specified directory.
"Could not load the application certificate."		
"Certificate is not trusted."	The server certificate is not on the list of trusted certificates.	Have the administrator copy the relevant certificate from "C:\ProgramData\Siemens\Automation\SIMIT\8.0\PKI\rejected\certs" to "C:\ProgramData\Siemens \Automation\SIMIT\8.0\PKI\trusted \certs".

OPC UA server: Error messages and remedies

Error message	Cause	Remedy
"Socket was closed gracefully."	The OPC UA server does not trust the SIMIT OPC UA client.	Add the OPC UA client to the list of trusted clients.
"Error received from remote host."		

2.9.6.8 Properties of the OPC UA client coupling

- **Time slice**
  Specifies the cycle in which the coupling exchanges data. The assignment of absolute cycle times to the 8 possible time slices applies to the entire project. The default is time slice 2, which corresponds to a cycle of 100 ms in the standard setting.

  Note

  The time slice with the lowest cycle time always has the highest priority, regardless of the numbering.

- **OPC UA server URL**
  Specifies the address at which the OPC UA server can be accessed. The following addresses are possible:
  - URL including port
    Port 4840 is assumed by default if you do not specify a port. This port uses the service "Local Discovery Server".
  - Server name
  - IP address
- **End point**
  Specifies the OPC UA server and transfer mode.
  - Protocols supported: OPC UA binary protocol (communication profile "UA-TCP UA-SC UA Binary")
  - Security policies supported: Basic256Sha256, Basic256, Basic128Rsa15, None
  - Security modes supported: None, Sign, Sign&Encrypt
- **User authentication**
  When user authentication is enabled, the user and password are requested by SIMIT. This information is stored user-specific and encrypted for connection establishment with the OPC UA server.
- **Name space URI**
  Specifies the scope of the signals provided by the selected OPC UA server. An OPC UA server can provide multiple name spaces. To use signals from multiple name spaces, create a separate OPC UA client coupling for each name space.
- **Status display**
  Specifies the name of the integer output signal that provides the status of the connected OPC server while simulation is running. You can use this output signals to evaluate the server status in your SIMIT project. One output signal is generated for each OPC UA client coupling.

See also

Configuring an OPC UA client coupling (Page 182)

Status of OPC servers (Page 169)

2.9.6.9 Operating principle of the SIMIT gPROMS coupling

Introduction

The function of the SIMIT gPROMS coupling is based on the SIMIT OPC UA server coupling. All properties and functionalities of the OPC UA server coupling are thus also available in this coupling.

In addition, this coupling also features the separation of the actual server functionality, which is executed as a Windows service, from the portion of the coupling running within the SIMIT application.

Note

In contrast to the OPC UA server coupling, the service assigned to the gPROMS coupling is not started automatically together with the simulation PC.

Behavior of the SIMIT gPROMS coupling

The gPROMS coupling makes available all configured signals as OPC UA signals, to which a correspondingly configured PSE/gPROMS® gFPI interface then connects.

The service assigned to the gPROMS couplings connects to the SIMIT RCI interface at runtime. As a result, in addition to all properties and functionalities of the OPC UA server coupling, this special coupling supports the following advanced functions:

- **Management of PSE/gPROMS® snapshot data**
  If the currently open SIMIT project contains valid configured gPROMS couplings, the following RCI commands are executed via the associated Windows service:
  - DeleteSnapshot
  - DeleteSnapshotFolder
  - RenameSnapshot
  - RenameSnapshotFolder
  - CopySnapshot
  - CopySnapshotFolder

  These commands are executed independent of the current status of the simulation.

  Note

  The consistency of the managed data is never checked by the gPROMS coupling.

  A notice is merely generated when the execution fails due to directory or file conflicts.

- **Start of PSE/gPROMS® process models**
  The gPROMS coupling supports the following two modes of operation:
  - Offline or Engineering mode: The user arrives at this mode when the "Start gPROMS simulation" option is not enabled. In this mode, the user can start a simulation in the PSE/gPROMS® Process Builder in parallel with a SIMIT simulation and connect to the OPC UA signals.
  - Online or gO:Run mode: The user arrives at this mode when the "Start gPROMS simulation" option is enabled. The service assigned to the gPROMS couplings starts the PSE/gPROMS® process models configured in the coupling at runtime. When simulation ends, the service also closes the associated instances of the PSE/gPROMS® process models.

  Note

  Export the configured PSE/gPROMS® process models in the right format beforehand.

  No check is made of the configuration for correctness or completeness.

  Notices are merely generated later in the simulation if a node of the gPROMS coupling was unable to be executed.

- **Control of PSE/gPROMS® process models**
  The simulation commands received via the command interface are transferred to marked OPC UA signals (Input), which are used only for control.
  The successful execution of the simulation commands is also acknowledged using specially created signals.
  This handshake mechanism allows the user to transfer the following commands, which are available via the SIMIT user interface, to the connected PSE/gPROMS® process models.
  - Open

- Close
- Init
- Reset
- Run
- CreateSnapshot
- LoadSnapshot
- SetSpeed

Note

The observance of the handshake mechanism is monitored by timeout conditions that, if exceeded, trigger a notification to the user.

If, however, the intended execution is not guaranteed due to an incorrect configuration of the gPROMS coupling or the gFPI within the PSE/gPROMS® process model, a restart of the SIMIT application may be necessary.

2.9.6.10 Enabling a gPROMS co-simulation

Requirement

Besides the installation requirements for SIMIT SP, additional boundary conditions apply, which only allow use of the gPROMS co-simulation.

- A Windows 64-bit operating system is installed on the simulation PC.
- SIMIT is installed.
- PSE/gPROMS® ProcessBuilder software is installed.

Note

Read the security notes in the preface.

Changes within the Windows Service Manager can affect the stability of your system!

If necessary, contact your system administrator before performing the steps described below.

Procedure

1. Open the "Windows Service Manager", for example, by entering "services.msc" in the system prompt and confirming with "Enter".
2. Select the "Siemens.Simit.OpcUaServer.gPROMSServerService" service.
3. Select "Properties" in the shortcut menu.
   The "Properties of Siemens.Simit.OpcUaServer.gPROMSServerService" window opens.
4. Activate the service using the "Start" button.

2.9.6.11 Configuring a SIMIT gPROMS coupling

Requirement

- Project is open in SIMIT.

Procedure

To create a SIMIT gPROMS coupling, follow these steps:

1. Select the "New coupling" command from the "Couplings" shortcut menu in the project tree.
2. Activate the coupling type "gPROMS".

Result

The SIMIT gPROMS coupling is configured. The following options are available for adding input and output signals:

- Entering signals manually
- Importing a signal table
- Importing a spreadsheet (Excel) file

Note

Besides the above-named formats for importing (signal table and spreadsheet), which can also be exported, an export in json format is also possible.

2.9.6.12 Properties of the gPROMS coupling

Once the coupling is opened, the coupling editor appears in the work area. You can define the following properties in the property view:

- **Path to gPROMS workspace**
  Specifies the path to the work area that is currently being used by the PSE/gPROMS® software or to which the model software was exported.

  Note

  For the exact procedure for modeling in PSE/gPROMS® and the aforementioned export, refer to the associated (ProcessBuilder) documentation.

- **Simulation model name**
  Name that was assigned to the simulation model during the export.

- **Password**
  Password that was assigned to the simulation model during the export.

  Note

  Because the entered password is only checked at runtime (Open simulation), the password must be entered twice for confirmation at this point.

- Start gPROMS simulation
  Specifies whether the PSE/gPROMS® process model present in the workspace is also executed with the gO:Run application at runtime of the SIMIT simulation model.

  Note

  Do not select this option if the PSE/gPROMS® process model is to be started from the ProcessBuilder environment.

2.9.6.13 Additional properties of the signals of a gPROMS coupling

Once the coupling is opened, the coupling editor appears in the work area. You can define the following properties in the property view:

- **Model name**
  This signal is accessed within the PSE/gPROMS® modeling using this name.
- **Client tag**
  The assignment of the signal (server tag) to this client tag, which is used within the PSE/gPROMS® model, is made within the gFPI interface.
- **RCI cmd type**
  The meaning of this parameter is dependent on the signal direction:
  - Input:
    Assignment of the RCI command that this signal activates at the time of execution. As a result of this, execution of the command is started within the PSE/gPROMS® gFPI.
  - Output:
    Assignment of the RCI command that is acknowledged when this signal arrives.

2.9.6.14 Certificates

Introduction

For data exchange over OPC UA, certificates confirm the identity of the connection partner. The certificates are automatically exchanged by the OPC UA client and OPC UA server when a connection is first established. Before each subsequent connection, the system checks whether the certificates are still valid.

Storage of certificates

The following directory structure is created in the "C:\ProgramData\Siemens\Automation\SIMIT \8.0\PKI\" directory during SIMIT installation. Within this directory structure, members of the "Administrators" group and the "SYSTEM" users have full access.

- "issuers": Contains certificates required for verifying CA certificates.
  Access rights:
  - "Users" group: Read
- "own": Contains the "Application Instance Certificate" and the private key of the OPC UA client, which is generated during installation. The OPC UA server reads this directory when a connection is first established. The private key is only generated once and is not overwritten when the software is updated.
  Access rights:
  - "Users" group: No access in the \private subdirectory
  - "Users" group: Read in \certs subdirectory
- "trusted\certs": Contains certificates of OPC UA servers with which the OPC UA client can exchange data.
  Access rights:
  - "Users" group: Read
- "rejected\certs": Contains rejected OPC UA server certificates.
  Access rights:
  - "Users" group: Write

Notes on certificate exchange

For security reasons, new connections to an OPC UA server are also rejected. A new connection must be confirmed by the administrator both on the OPC UA client and on the OPC UA server.

- OPC UA server: The OPC UA client must be included in the list of "trusted clients".
- OPC UA client: Certificates sent by the OPC server are saved in the "rejected\certs" folder the first time a connection is established. You then need to copy the certificates manually to the directory "trusted\certs".

By default, only members of the "Administrators" group have access rights to the folders "own \private" and "trusted\certs". You have the following options if another user wants to set up an OPC UA client coupling:

- The administrator sets up the necessary access rights for the user in question.
- The administrator grants users read privileges to the "own\private" folder and copies the certificates stored under "rejected\certs" to the "trusted\certs" folder.

Following one of these two actions, the user can then establish the connection to the OPC UA server.

For security reasons, these privileges are reset to the default settings on every installation and may need to be adjusted again.

See also

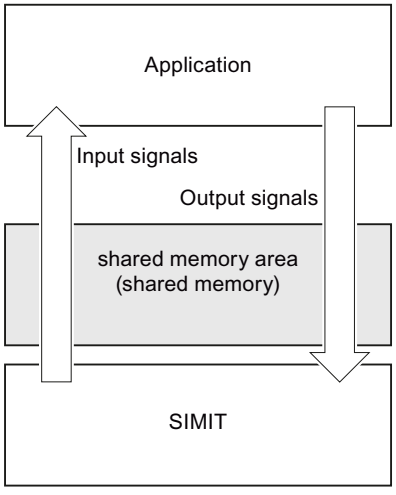
Configuring an OPC UA client coupling (Page 182)

2.10 Shared memory coupling

2.10.1 How the SHM coupling works

SIMIT uses the Shared Memory coupling, SHM coupling for short, to communicate with any number of other applications over shared memory (SHM). This coupling is all-purpose and high-performance.

Input signals are the signals that are written by SIMIT to the memory area and output signals are the signals that are read by SIMIT from the memory area.



Note

Access to the shared memory area is not limited to a particular user. Each application that runs on this PC has access to the shared memory area.

2.10.1.1 Accessing the memory area

If multiple independent processes access the same memory area, access must be synchronized to ensure consistent values. A mutex is used in the SHM coupling as the synchronization object.

In each simulation time slice, SIMIT writes and reads all the input and output signals defined in the SHM coupling and blocks the mutex while it does so. All other applications that access this memory area should do the same.

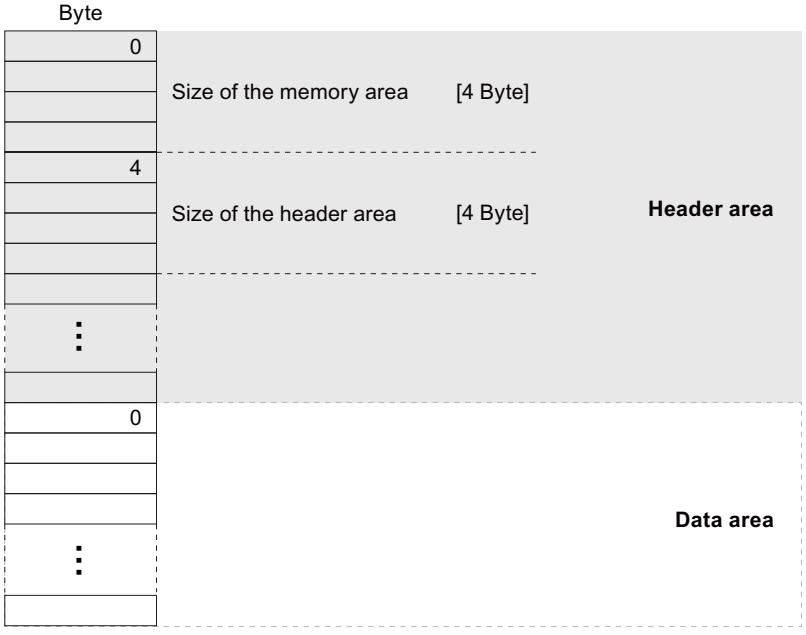
Note

Each application that is connected to SIMIT over an SHM coupling should minimize the time for which it blocks the mutex so as not to block access to the shared memory area by SIMIT and other applications for longer than necessary.

2.10.1.2 Structure of the memory area

Header area and data area

The memory area is divided into a header area and a data area. The header area size is at least 8 bytes. The first 4 bytes of the header area contain the size of the total memory area, and the subsequent 4 bytes the size of the header area. Little-endian format applies for both values.



Structure of the data area

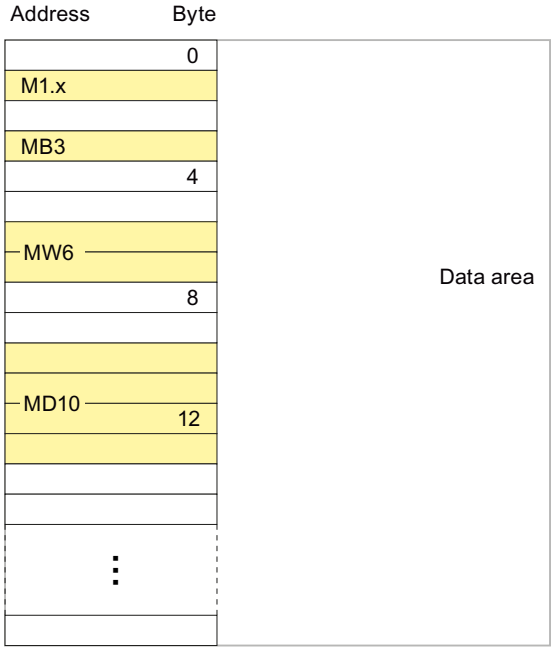
SIMIT addresses the data area byte-by-byte in the same way as for addressing the I/O area of SIMATIC automation systems. Each signal in the SHM coupling is linked to a unique address in the data area. As a signal at one address in the data area can only be defined either as an input signal or as an output signal, the same address cannot be assigned to both and input and output signal.

The input and output signals are mapped to a shared memory area. No overlaps between input and output signals are permitted. Overlapping input and output signals are detected in the consistency check and displayed as inconsistencies.

A signal occupies 1, 2 or 4 bytes at its address in the data area depending on its data type:

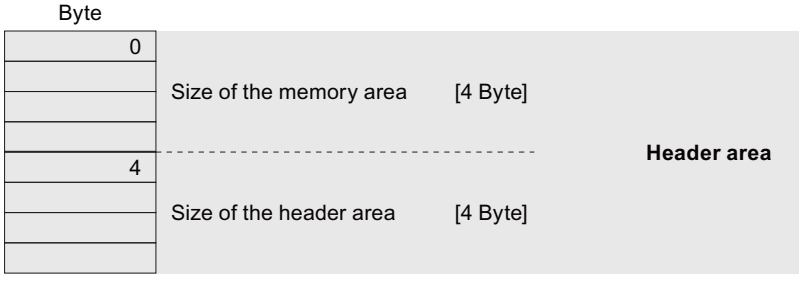
- 1 byte for the data types BOOL and BYTE,
- 2 bytes for the data types WORD and INT and
- 4 bytes for the data types DWORD, DINT and REAL.

The figure below shows an example of addressing for various different data types:

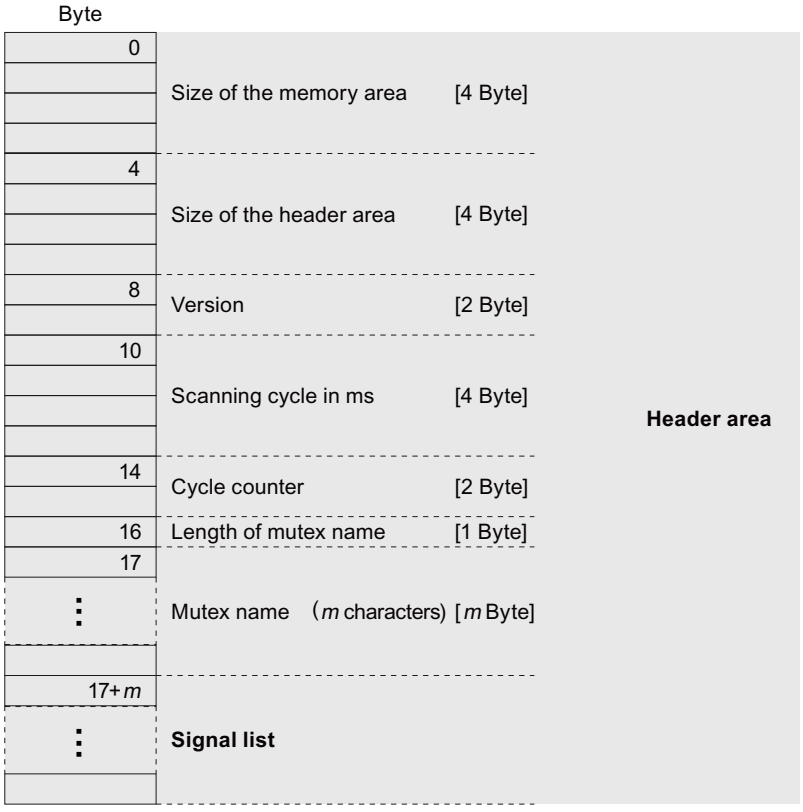


Structure of the header area

The minimum header area size is 8 bytes. The first 4 bytes of the header area contain the size of the total memory area, and the subsequent 4 bytes the size of the header area.



If SIMIT creates the shared memory area, SIMIT can optionally add additional variables and a list of signals to the header area. This information can then be used by applications to configure their access to the data area.



The version identifies the memory structure. As the structure defined here has the version identifier "0", you will always find a value of "0" entered here. If the structure of the memory area is altered, the version is changed accordingly.

The predefined sampling time slice for the SHM coupling is entered in milliseconds (ms) as an integer value.

In each cycle of the SHM coupling, the integer value of the cycle counter is incremented by 1.

The length of the mutex name, which means the number *m* of characters in the mutex name, is stored in byte 16 of the header area. The mutex name is saved in the following header area from byte 17.

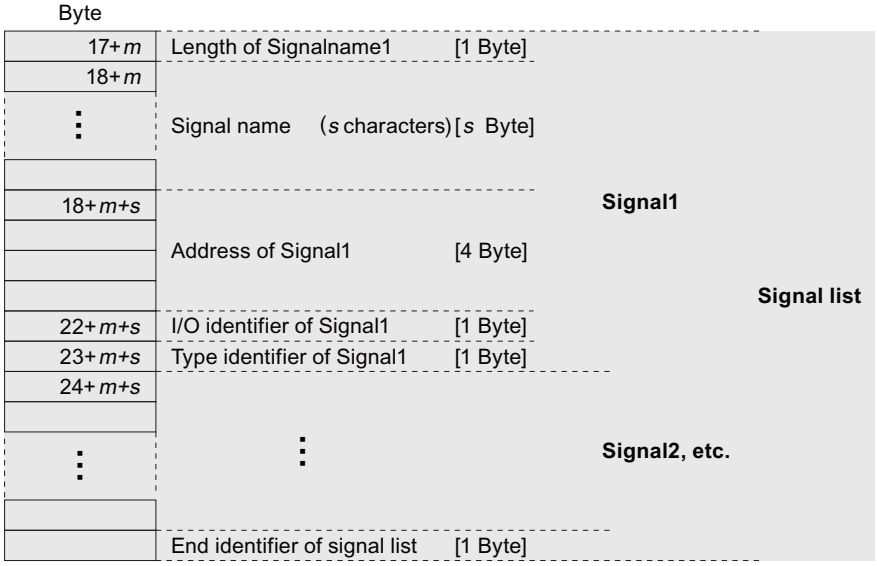
The signal list created from byte 17+*m* onwards in the header provides information about the signals in the data area. The following variables are specified for each signal:

- Length of the signal name, i.e. the number of characters in the signal name
- Signal name
- Address of the signal, i.e. the signal offset in the data area
- I/O identifier, the identifier that shows whether a signal is an input signal (0) or output signal (1)
- Type identifier for the data type

Table 2-3 Signal type identifiers

Type identifier	Meaning (data type)
0	BOOL, bit address 0
1	BOOL, bit address 1
...	...
7	BOOL, bit address 7
8	BYTE
9	WORD
10	INT
11	DWORD
12	DINT
13	REAL

The figure below shows the structure of the signal list in the header area:



This signal list ends with an end identifier with a value of "0".

2.10.1.3 Creating the memory area

The shared memory area can be created either by SIMIT or by another application that is connected to SIMIT via the memory area. SIMIT responds accordingly when a simulation containing an SHM coupling is started.

If the shared memory area was created by an application, i.e. already exists when the simulation is started in SIMIT, the memory area is opened by SIMIT. SIMIT only connects to the memory area if the size of the data area corresponds to the size of the address area defined by the input/output signals in the SHM coupling. Otherwise, an error message is output and SIMIT does not connect to the memory area.

If there is no shared memory area when SIMIT is started, it is created by SIMIT. SIMIT enters the size of the entire memory area and the header area into the header area. The size of the data area is determined by the highest address of the signals defined in the SHM coupling. Optionally,

other sizes and a list of the signals can be entered in the header area by SIMIT. You can find additional information in section: Structure of the memory area (Page 192).

2.10.2 Configuring the SHM coupling

2.10.2.1 Creating an SHM coupling

To create an SHM coupling, select the option "SHM" in the "Selection" dialog box.

Note

A maximum of 32 couplings of the SHM and Plant Simulation types can be created together in one project.

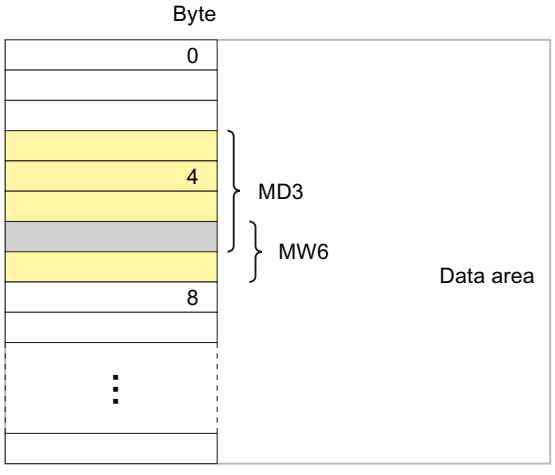
2.10.2.2 Configuring the signals in the SHM coupling

The options for entering input/output signals for the coupling are as follows:

- Enter the signals manually in the coupling editor.
- Import the symbol table. You can find additional information on this in section: "Importing signal properties" dialog box (Page 988).

SIMIT reads the output signals from and writes the input signals to the shared memory area. Access takes place cyclically in the cycle defined for the coupling. One byte of the memory area must be uniquely assigned to either an input signal or an output signal. Each input signal must be uniquely represented in the data area and the data areas of different input signals must not overlap.

The figure below shows an example with 2 input signals, MD3 and MW6, that overlap in byte 6:



The SIMIT consistency check reports overlapping signals as errors.

Note

As the smallest unit of the memory area that can be addressed is 1 byte, 1 byte can only be assigned either to an input signal or to an output signal. Binary signals with the same byte address can therefore only be either all binary input signals or all binary output signals.

SIMIT also accesses the memory area with byte-by-byte rather than bit-by-bit write access. 1 bit in one byte of the memory area for which no binary input signal is defined in the SHM coupling is set to "0" upon SIMIT write access to the memory area.

The address name of a signal starts with "M" for "Memory" and then contains the data type and address. Based on the address notation in SIMATIC automation systems, the data types set out in the table below can be used.

Table 2-4 Definition of data types

Data type	Variable	Notation	Value range
BOOL	1 bit	M<byte>.<bit>	True/False
BYTE	1 byte (8 bits)	MB<byte>	0 ... 255 or –128 ... 127
WORD	2 bytes	MW<byte>	0 ... 65,535
INT	2 bytes	MW<byte>	–32,768 ... 32,767
DWORD	4 bytes	MD<byte>	0 ... 4,294,967,295
DINT	4 bytes	MD<byte>	–2,147,483,648 ... 2,147,483,647
REAL	4 bytes	MD<byte>	$\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$

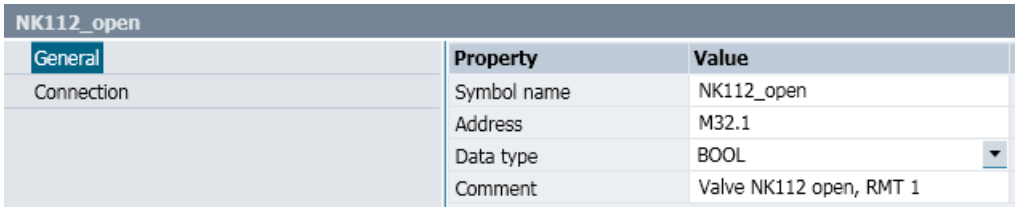
The following applies for mapping the integer signals in SIMIT simulation projects to the data types in the memory area:

- The least significant byte (LSB) of the signal is used for the BYTE data type.
- The WORD and DWORD data types are unsigned and the INT and DINT data types are signed. The values are limited to the value ranges specified in the table above.

The structure of a floating-point number is governed by the "IEEE Standard for Binary Floating Point Arithmetic" (ANSI/IEEE Std 754-1985).

2.10.2.3 Signal properties in the SHM coupling

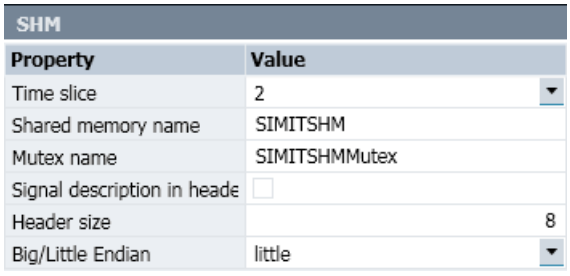
The properties of a signal are shown in the individual columns of the coupling editor and in the property view.



- **Symbol name**
  The signal is identified by this name in SIMIT.
- **Address**
  Signal values are stored in the data area of the shared memory area under this byte address. The address "0" corresponds to the first byte after the header area.
- **Data type**
  The data type specifies the space a signal occupies in the data area and how the signal values saved there are to be interpreted. The options are as follows:
  - Logical value
  - Signed integer value
  - Unsigned value
  - Floating-point number
- **Comment**
  The comment is used to document the signal. It is not evaluated.

2.10.2.4 Properties of the SHM coupling

Once the coupling is opened, the coupling editor appears in the work area. You can define the following properties in the property view:



- **Time slice**
  Here you set the cycle at which the coupling exchanges data. The assignment of absolute cycle times to the 8 available time slices is valid for the entire project. Time slice 2 is the default, corresponding to a cycle of 100 ms.

  Note

  The time slice with the lowest cycle time always has the highest priority, regardless of the numbering.

- **Shared Memory name**
  Here you enter the name with which the shared memory area can be addressed.
- **Mutex name**
  Here you enter the name of the mutex for synchronizing access to the shared memory area.

- **Signal description in header**
  This option allows you to choose whether or not SIMIT is to create an extended header area. You can find additional information on this in section: Structure of the header area (Page 193).
- **Header size**
  Here you enter the size of the header area in bytes. The value may be freely chosen, but must be at least 8 bytes. If SIMIT creates the shared memory area, a header area with the specified size is created.

Note

If you activate the "Signal description in header" option, the size of the header area is determined by SIMIT based on the input/output signals in the coupling. The "Header size" property is not editable in this case.

- **Big/Little Endian**
  This property determines the byte order in which values of the WORD, INT, DWORD and DINT data types are encoded in the data area.

Setting	Byte order
Big endian	The most significant byte is stored first, which means at the lowest memory address.
Little endian	The least significant byte is stored first, which means at the lowest memory address.

2.10.2.5 Importing and exporting signals

SIMIT can store coupling content in txt format; this format contains SIMATIC addresses and information about scaling. Because this kind of information does not exist in SHM couplings, this format is only suited to a limited extent. However, you can still export import signal tables in txt format; only relevant information is available or taken into consideration in the SHM coupling.

SIMATIC symbol tables can also be imported in asc, seq and xlsx format as well as txt format. Especially in this case, ensure that addresses for inputs and outputs in the SHM coupling are not allowed to overlap.

Note

If you open the signal table for editing in Excel, all cells must be formatted as "text" so that Excel does not make any unwanted format conversions.

You can find additional information on this in the following sections:

- Signal table (Page 223)
- "Importing signal properties" dialog box (Page 988)
- "Exporting signal properties" dialog box (Page 991)

2.11 Plant Simulation

The "Plant Simulation" coupling corresponds to the Shared Memory coupling in terms of handling and functionality.

You can find more information on the coupling under Shared memory coupling (Page 191).

2.12 PRODAVE coupling

2.12.1 How the PRODAVE coupling works

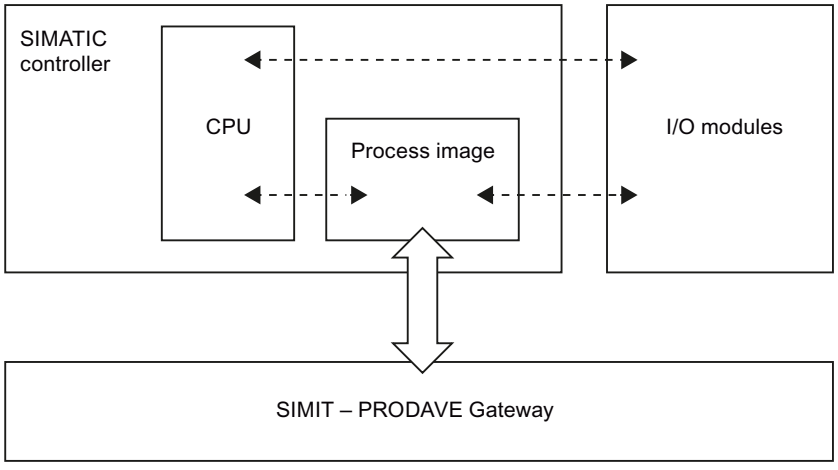
SIMIT uses the PRODAVE coupling to communicate with a SIMATIC controller over the MPI interface. For Ethernet-capable controllers, the connection can also be made via Ethernet. An Ethernet connection performs somewhat better than a connection via MPI.

In SIMATIC Manager, set the PG interface to the selected connection type.

Note

To use the PRODAVE coupling you also need version 6.2 of the SIMATIC PRODAVE software. This software is not included in the SIMIT product package. You may also need additional hardware, such as MPI adapters or connecting cables. These also are not included in the SIMIT product package.

The PRODAVE coupling does not simulate any SIMATIC I/Os. SIMIT directly addresses the process image of a SIMATIC controller via the PRODAVE coupling. The addresses that are actually addressable depend on the size of the CPU process image and on what was defined in your hardware configuration.



Note

Do not use any I/O modules connected to your controller that use the same address ranges as the PRODAVE coupling. They will otherwise compete for access to the process image.

Note

If you have configured I/O modules in HW Config, you can access the address ranges from the PRODAVE coupling provided the configured modules are not actually connected. This will cause access errors in the CPU, which you can catch by creating the corresponding OBs in the control program. Because the PRODAVE coupling only has access to the process image, you cannot access any simulated signals that are addressed in the CPU as I/O signals (*PIW, PQW*). Some SIMATIC controllers such as CPU 313C have fixed connections to I/O modules whose address ranges cannot be changed. These address ranges cannot be accessed via the PRODAVE coupling.

2.12.2 Configuring the PRODAVE coupling

2.12.2.1 Creating a PRODAVE coupling

To create an PRODAVE coupling, select the option "PRODAVE" in the "Selection" dialog box.

2.12.2.2 Editing signals in the PRODAVE coupling

The options for entering input/output signals for the coupling are as follows:

- You enter the signals manually in the coupling editor.
- You import the symbol table from your SIMATIC project. You can find additional information in section: "Importing signal properties" dialog box (Page 988).

2.12.2.3 Properties of the PRODAVE coupling

Once the coupling is opened, the coupling editor appears in the work area. You can define the following properties in the property view:

PRODAVE	
Property	**Value**
Time slice	2
Mnemonic	E/A
CPU slot	2
Access mode	MPI
MPI address	2

- **Time slice**
  The cycle in which the coupling exchanges data is set here. The assignment of absolute cycle times to the 8 possible time slices applies to the entire project. Time slice 2, which corresponds to a cycle of 100 ms, is preset.

  Note

  The time slice with the lowest cycle time always has the highest priority, regardless of the numbering.

- **Mnemonics**
  Here, you select whether the international (I/Q) or German (E/A) mnemonics are to be used for the inputs and outputs.
- **CPU slot**
  Enter the slot number of your SIMATIC CPU here.
- **Access mode**
  Here, you select whether to you want to use MPI (cable or adapter) or IP (Ethernet) access.
- **MPI address / IP address**
  Enter the MPI number / IP address of the SIMATIC CPU here.

Additional functions

Copying and pasting scaling

You can find additional information in section: Transferring scaling to another signal (Page 239).

Importing signal properties

You can find additional information in section: "Importing signal properties" dialog box (Page 988).

Exporting signal properties

You can find additional information in section: "Exporting signal properties" dialog box (Page 991).

2.12.2.4 Importing the signal properties

Import the signal properties by clicking the "" symbol in the coupling editor.
The "Import of signal properties" dialog box opens. You can make the import settings in this dialog box.

You can find additional information in the section: "Importing signal properties" dialog box (Page 988).

2.12.2.5 Exporting the signal properties

Click on the "" symbol in the coupling editor to export the signal properties.
The "Exporting signal properties" dialog box opens. You can make the export settings in this dialog box.

See also

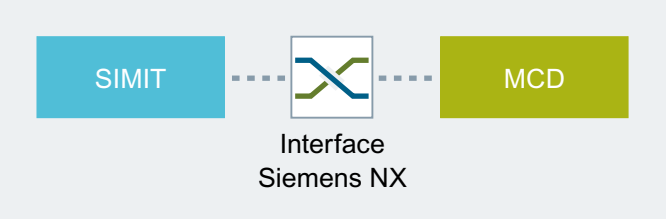
"Exporting signal properties" dialog box (Page 991)

2.13 Mechatronics Concept Designer coupling

2.13.1 How the Mechatronics Concept Designer coupling works

Method of operation

With the MCD coupling, SIMIT communicates with a Mechatronics Concept Designer application over a software interface of SIEMENS NX. SIMIT uses the MCD coupling for cyclic exchange of data from the signal pool and the physical objects of the coupled MCD applications.



The following applications must be installed on a PC to use the MCD coupling:

- SIMIT
- Mechatronics Concept Designer

To exchange data with the MCD application, this needs to be allowed by the MCD application. The signals that are enabled for SIMIT are also specified here.

For this purpose, MCD provides an "Allow connection" button. This button retains its state once it has been pressed once, meaning that the status "Allow connection" is retained until you press the button again and return it to its original state or until you end the MCD instance.

Note

If this button is selected in multiple running MCD instances, the connection is accepted for simulation with any of these MCD instances.

In MCD, you can load component assemblies into different tabs. For simulation, SIMIT uses the component assembly from the tab that is selected when simulation starts. Re-synchronization of the input and output signals does not take place. Exchange of simulation data only takes place for signals whose name and type match in SIMIT and MCD.

2.13.2 Usage scenarios of the Mechatronics Concept Designer coupling

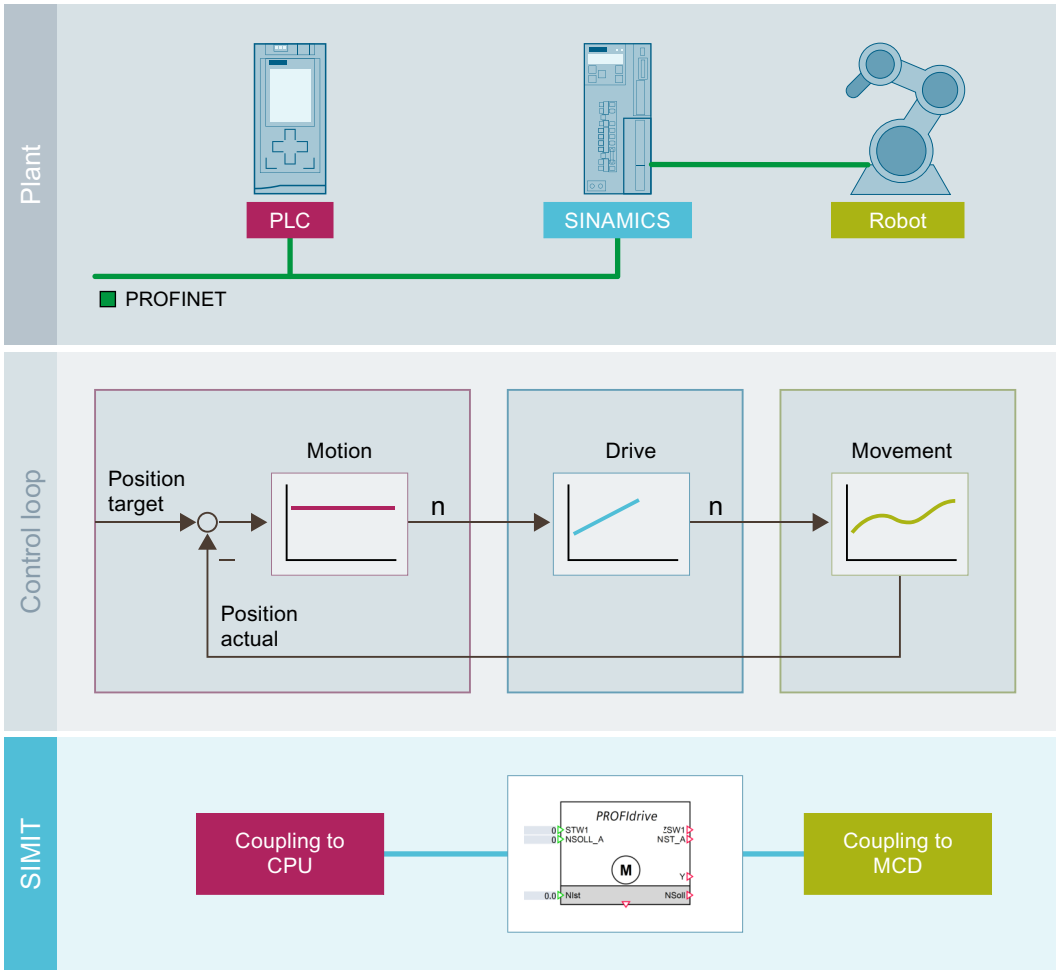
Operating principle

With the "Mechatronics Concept Designer", you can expand your behavior modeling in SIMIT by a physical model. The aim of a total simulation with the MCD is to check motion control on a machine which is specified by a technological controller.

Using SIMIT with a "Mechatronics Concept Designer" application requires the following couplings in SIMIT:

- Couplings to a controller of the SIMATIC family
- MCD coupling

The figure below shows the simulation of a control schematically using a position change of a robot axis as an example:



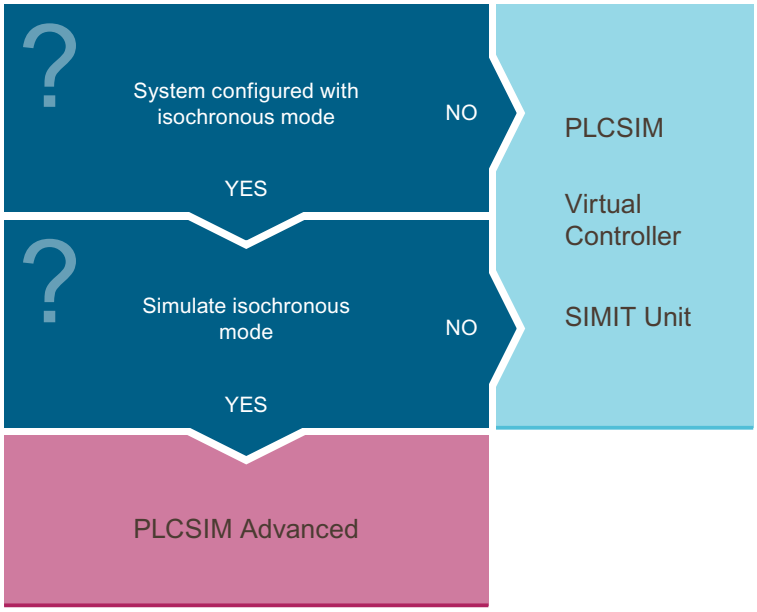
The "Coupling to CPU" transfers the control command to the PROFIdrive component.

You can model the behavior of a drive in SIMIT with the PROFIdrive components from the SIMIT component library. For this purpose, the drive needs to have an interface corresponding to the PROFIdrvive standard.

The PROFIdrive component calculates the target speed based on the control command. The target speed is transferred over the MCD coupling to the "Mechatronics Concept Designer" application. The "Mechatronics Concept Designer" application contains the CAD model of the robot and its physical properties. Using a physics engine for modeling the physical conditions, "Mechatronics Concept Designer" calculates the actual movement of the component from the specified target speed. Then the Mechatronics Concept Designer application returns the calculated actual position. This is made available as position tracking in the technological controller in the PLC.

Isochronous mode

If you have configured a system with isochronism and if you want to simulate the isochronous mode, you have to use a PLCSIM Advanced and use isochronous mode in SIMIT.



PLCSIM Advanced, SIMIT and MCD correspond to PLC, SINAMICS and robot in the plant.

Simulate Mechatronics Concept Designer application with configured isochronous mode

1. Creating a SIMIT project
2. Select "Isochronous" as the operating mode in the project properties.
3. Configuring a time slice with the cycle time
4. Configuring a PLCSIM Advanced coupling
5. Set PLCSIM Advanced coupling to "Isochronous" mode.
6. Set Mechatronics Concept Designer coupling to "Isochronous" mode.
7. Configuring an MCD coupling
8. Configuring a PROFIdrive component

9. Assigning components to the time slice with the set cycle time.
10. Interconnecting the PROFIdrive component with signals from both couplings

2.13.3 Configuring Mechatronics Concept Designer coupling

2.13.3.1 Creating the Mechatronics Concept Designer coupling

Introduction

In a SIMIT project, you can create a coupling of the type "Mechatronics Concept Designer".

Requirement

- A project is open in SIMIT.
- Mechatronics Concept Designer Version 12.0.2 or higher is installed on the SIMIT PC.

Procedure

To create a coupling of the type "Mechatronics Concept Designer", follow these steps:

1. Add a new coupling in the project navigation under "Couplings".
   The "Selection" dialog box opens.
2. Activate the coupling type "Mechatronics Concept Designer".

Result

The coupling of the type "Mechatronics Concept Designer" is created in the project navigation.

2.13.3.2 Importing signals

Requirement

- The project is opened in SIMIT.
- A coupling of the type "Mechatronics Concept Designer" has been created.
- Your "Mechatronics Concept Designer" application is open.

Procedure

To import a "Mechatronics Concept Designer" application and its signals, follow these steps:

1. Double-click the coupling of the type "Mechatronics Concept Designer" in the project navigation.
   The coupling editor of the "Mechatronics Concept Designer" coupling opens.
2. Click the button "Receive signals from MCD".
   The "Receive signals from MCD" window opens.
3. Confirm sending of the signals in your "Mechatronics Concept Designer" application by pressing the "Send signals to SIMIT" button.

Result

The input and output lists of the "Mechatronics Concept Designer" coupling in SIMIT are now filled.

2.13.3.3 Editing signal properties in the Mechatronics Concept Designer coupling

Introduction

The coupling of the type "Mechatronics Concept Designer" permits a configuration of the physical unit of each exchanged signal. The physical unit configured in the "Mechatronics Concept Designer" coupling is valid for use within the simulation model in SIMIT. The physical unit of the same signal in the MCD application can certainly be different.

Requirement

- The project is opened in SIMIT.
- A coupling of the type "Mechatronics Concept Designer" has been created and access from SIMIT to the MCD application is possible.
- SIEMENS NX in version 12.0.2 or higher is installed on the SIMIT PC.

Procedure

To configure the physical unit of a signal of the "Mechatronics Concept Designer" coupling, follow these steps:

1. Double-click the coupling of the type "Mechatronics Concept Designer" in the project navigation.
   The coupling editor of the "Mechatronics Concept Designer" coupling opens.
2. Select the desired signal in the "Input/output signal" area of the coupling editor.
   The signal is highlighted.
3. In the "Unit" column, select the desired physical unit from the drop-down list.
4. Save the changes in the coupling editor.

Result

The physical unit of the desired signal of the "Mechatronics Concept Designer" coupling has been changed.

2.13.3.4 Configuring the isochronous Mechatronics Concept Designer coupling

Requirement

- A "Mechatronics Concept Designer" coupling has been created.
- "Isochronous" mode is set.
- A time slice with the configured cycle time from STEP 7 is configured.

Procedure

To configure an isochronous "Mechatronics Concept Designer" coupling, proceed as follows:

1. Double-click on the "Mechatronics Concept Designer" coupling in the project navigation under "Couplings".
2. Configure the "Mechatronics Concept Designer" coupling in the property view:
   - Select the time slice that is configured with the cycle time from STEP 7.
   - Select the "Isochronous" option.

Result

The isochronous "Mechatronics Concept Designer" coupling is configured.

2.13.3.5 Properties of the Mechatronics Concept Designer coupling

You can define the following properties in the property view:

Time slice

Here you set the cycle at which the coupling exchanges data. The assignment of absolute cycle times to the 8 available time slices is valid for the entire project. Time slice 2 is the default, corresponding to a cycle of 100 ms.

Note

The time slice with the lowest cycle time always has the highest priority, regardless of the numbering.

Isochronous

Specifies whether the MCD application is synchronized with the SIMIT model calculation.

If you set isochronous here, you must configure the set clock time in the selected time slice.

If you set isochronous here, you also have to set isochronous in the properties window in the Project Manager. You can find additional information on isochronous mode in the Operation modes section.

Include Physics Objects

Specifies whether signals of physical objects are also imported or only the signals that you created under the Signals folder in your "Mechatronics Concept Designer" application.

NX Operating mode

Shows the operating mode.

MCD Part

Shows the referenced MCD part file (main file of your MCD application).

Archive

Shows the path to the archive of your "Mechatronics Concept Designer" part file. If the field is empty, no archiving of your part file to the SIMIT project has taken place yet.

See also

Operating modes (Page 39)

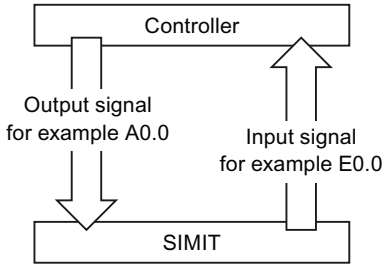
2.14 Editing the signals of a coupling

2.14.1 Signal basics

2.14.1.1 Coupling data direction

Each coupling in SIMIT defines signals that can be linked to simulation components. The terms "input" and "output" are from the perspective of the connected controller:

- An input signal is a signal that is calculated or output by the simulation and read by the controller.
- An output signal is a signal that is output by the controller and used as an input by the simulation.



All signals contained in a coupling are listed in the coupling editor. Access to a signal from the coupling is with the symbol name or the absolute address, irrespective of how the signal is used.

2.14.1.2 Meaning of the coupling name

Signal names consist of a source and a name. For any coupling signal the name of the coupling itself defines the source, and the symbolic name or the absolute address of the signal defines the name of the signal.

Source	Name
SIMATIC 400(1)	FC111

Signal names therefore remain unique throughout the entire SIMIT project, even if the same symbol name or the same address occurs in several different couplings.

Note

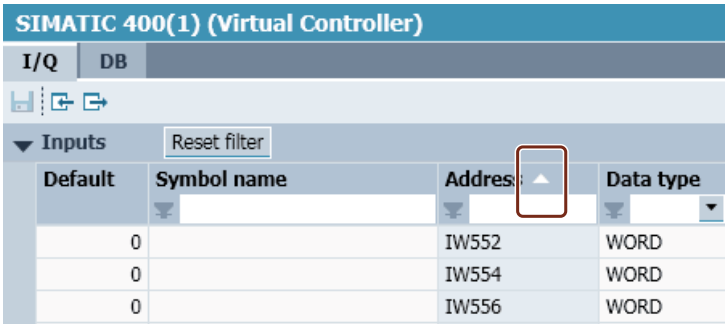
Each coupling within a SIMIT project must have a unique name.

For example, if you want to connect SIMIT to multiple PROFIBUS DP masters, create a separate coupling and assign a unique coupling name for each PROFIBUS DP master system. This ensures that all coupling signals are assigned to the correct coupling.

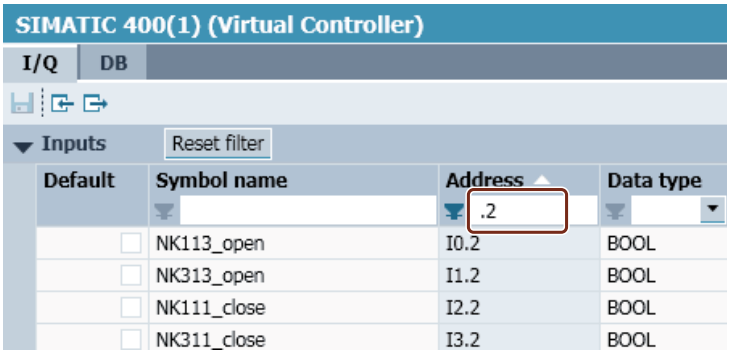
2.14.1.3 Sorting and filtering of signals in the coupling editor

Signals can be sorted and filtered by all properties shown in the individual columns. Follow these steps:

- Click the title of the column by which the signals are to be sorted.
  An arrow appears in the column title:



- Every additional click on the column header toggles between alphabetically ascending and descending sort order. The arrow changes its direction:
  - for sorting in ascending order
  - for sorting in descending order
- To further reduce the number of signals displayed, you can also set a filter in each column. In this case, only signals that meet all filter criteria are displayed.
  In the figure below, a text filter for the address of signals is set:



- Select a filter from a drop-down menu or enter a text.

2.14.1.4 Addressing signals

In a SIMATIC PLC, different mechanisms are used to access the process image (I/O) and the I/O addresses (PI/PQ). For SIMIT itself, this access difference is not relevant. To use SIMATIC couplings in SIMIT, you can either use *QW* or *PQW* or *IW* or *PIW* in the address of the I/O signal. Signals are exchanged as follows, depending on the type of coupling:

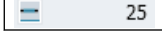
- The signal exchange with SIMIT in the SIMIT Unit coupling takes place over field device I/O signals, as in a real plant. For the controller, there is no difference to data exchange with real I/O. In the SIMATIC PLC, the I/Os or the process image can thus be accessed using the relevant mechanisms.
- In the case of the PRODAVE coupling, SIMIT only exchanges I/O signals with the process image of the controller. Access in the SIMATIC PLC to I/O addresses results in access errors.
- Think of the mechanisms for the PLCSIM coupling like those of the SIMIT Unit coupling: SIMIT communicates with PLCSIM on the basis of the I/O signals. The access mechanisms of the PLC are coordinated by PLCSIM.

2.14.1.5 Fixing signals in the coupling editor

Introduction

Signals can be set and "fixed" at a specific value. With fixation, the signal values currently pending are applied first to achieve a smooth switchover.

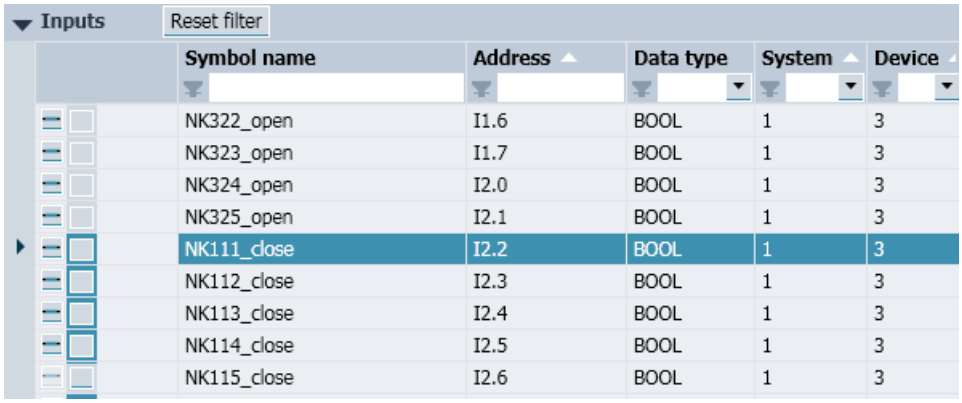
There is an automatically generated control for each signal to display fixation in the coupling editor:

	"Toggle switch with fixation" for signals of the data type "binary"
	"Digital input with fixation" for signals of the data types "analog" and "integer"

Note

Retaining a specific signal state is known as "forcing" in a SIMATIC controller. Fixing in SIMIT is a comparable function, but only applies within the simulation model and does not cause "forcing" in the SIMATIC controller.

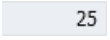
For each coupling signal, the controls with fixation are available in the first column after starting the simulation:



Symbol name	Address	Data type	System	Device
NK322_open	I1.6	BOOL	1	3
NK323_open	I1.7	BOOL	1	3
NK324_open	I2.0	BOOL	1	3
NK325_open	I2.1	BOOL	1	3
NK111_close	I2.2	BOOL	1	3
NK112_close	I2.3	BOOL	1	3
NK113_close	I2.4	BOOL	1	3
NK114_close	I2.5	BOOL	1	3
NK115_close	I2.6	BOOL	1	3

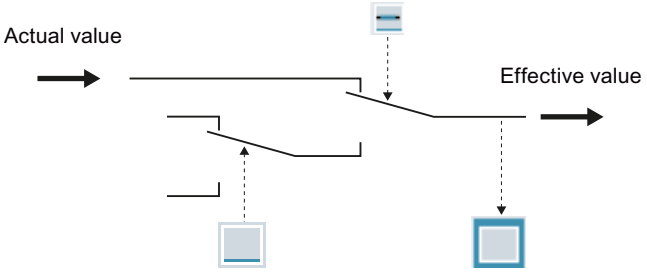
Principle of operation of the switch with fixing

The toggle switch with fixation for binary values combines three functions:

Symbol	Meaning
	Fixation switch is not active; the current value of the signal is displayed Fixation switch is active; the fixed value is displayed and can be used
	This symbol is only displayed with binary values: Value is "0" Value is "1"
	Input and display field for values of the data types "analog" and "integer"

If the fixation switch and/or the value cannot be operated, the corresponding symbol appears on a gray background, for example if the coupling signal is not used in the simulation model.

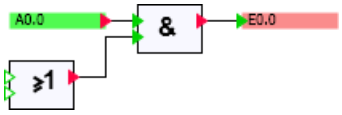
The figure below shows how a toggle switch with fixation works:



Override is basically a changeover, which either transfers the actual value in the coupling or changes it to a value that is input manually. It is possible both with input and with output signals i.e. all I/O signals can be specified.

Example of fixation

In the example below, two signals are configured in the coupling and used in a chart in the following way:



In the coupling you can see which value the controller outputs for the signal Q0.0.

Value output by the controller	Q0.0 in the coupling	Value used in the simulation model
0		0
1		1

Override allows you to control your simulation model so that it uses a different value from the one output by the I/O. To do this, click on the fixation switch for the signal *Q0.0*. This is then shown like this: . You can now use the toggle switch to directly enter the value to be used in your simulation model as follows:

Value output by the controller	Q0.0 in the coupling	Value used in the simulation model
0 or 1		1
0 or 1		0

The same applies to the input signals. For an input signal I0.0 you can first of all see what value the simulation model outputs to the controller:

Value calculated in the simulation model	I0.0 in the coupling	Value output to the controller
0		0
1		1

Using fixation, you can set the value that is output at the controller irrespective of the value calculated in the simulation model. Enter the default value using the fixation switch:

Value calculated in the simulation model	I0.0 in the coupling	Value output to the controller
0 or 1		1
0 or 1		0

This procedure applies in the same way to analog and integer signals. Only the default values are entered with a digital input, unlike the binary values. A digital display is used in place of the binary display.

See also

Fixing a signal (Page 237)

2.14.1.6 Mapping of SIMATIC data types in SIMIT

A signal in the SIMATIC coupling has one of the SIMATIC-specific data types *BOOL, BYTE, WORD, INT, DWORD, DINT* or *REAL*, which correspond to access to the address range of the PLC. When used as an I/O signal it has the data type binary, integer or analog. The table below shows the mapping of SIMATIC data types to the data types of the I/O signal.

Table 2-5 Mapping of SIMATIC data types to signal data types

SIMATIC data type	SIMATIC value range	SIMIT data type	Mapping to SIMATIC data types
BOOL	true, false	Binary	true, false
BYTE	0 .. 255	Integer	[–128 .. 255]
WORD (without scaling)	0 .. 65535	Integer	[–32768 .. 65535]
INT	–32768 .. 32767	Integer	[–32768 .. 65535]
DWORD	0 .. 4294967295	Integer	[–2147483648 .. 4294967295]
DINT	–2147483648 .. 2147483647	Integer	[–2147483648 .. 4294967295]
REAL	$\pm 1.175495 \times 10^{-38}$ $\pm 3.402823 \times 10^{38}$	Analog	$\pm 1.175495 \times 10^{-38}$ $\pm 3.402823 \times 10^{38}$

Values that are read in the SIMIT coupling, are applied to the SIMIT data types according to the value range of the SIMATIC data types listed above.

All values output by SIMIT through the coupling that are within the SIMATIC value range are output unchanged by SIMIT. Values outside this range are limited to the interval specified in the last column and mapped to the data width of the corresponding SIMATIC data type. Negative figures are saved as two's complement. For example, the figure –1 is transferred as a byte with the same bit pattern as the figure 255.

Scaled signals are dealt with slightly differently. Raw values that are read into the SIMIT coupling are limited to the value range of the SIMATIC and then converted to physical values.

Values output by SIMIT through the coupling are converted to raw values and then limited to the SIMATIC value range:

SIMATIC data type	SIMATIC value range	SIMIT data type	Mapping to SIMATIC data types
WORD (bipolar scaling)	[–32768 .. 32767]	Analog	[–32768 .. 32767]
WORD (unipolar scaling)	[–32768 .. 32767]	Analog	[–32768 .. 32767]
WORD (user-specific scaling)	[–32768 .. 32767]	Analog	[–32768 .. 32767]

You can find additional information on this in section: Scaling analog signals (Page 218).

2.14.1.7 Access to a data record or memory area

Access to data records in the distributed I/Os or to the memory area of a SIMATIC controller is possible with the following coupling types:

Coupling type	Access mode	Component
SIMIT Unit Virtual controller PLCSIM Advanced	Read/write data record in distributed I/Os	ReadRecordHandler WriteRecordHandler
Virtual controller PLCSIM PRODAVE	Read/write bytes from bit memory address area	ReadMemory WriteMemory
	Read / write bytes from data block	ReadDataBlock WriteDataBlock

This access is not carried out by cyclic communication between the controller and signals that are listed in the coupling editor, but through components which are triggered to start a read or write process. The required component types can be found in the basic library in the directory *COMMUNICATION* | *SIMATIC*.

You can easily link the component with corresponding coupling by entering, in the properties window of the unit connector, the name of the coupling that you want to access with that component. Alternatively, you can create the unit connector using drag & drop by dragging the module to be addressed from the property view of the coupling to the chart.

To use this access method for a coupling, you must have already saved the coupling. Open the coupling in the editor, define, for example, an input or output signal and then save the coupling.

Access to the memory area of a controller

Enter the coupling name in the property window of the unit connector. Connect the unit connector with the "Gateway" input of the component.

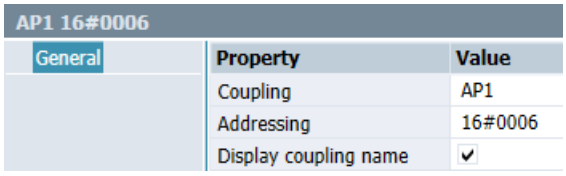
Access to a data record in the distributed I/Os

Enter the coupling name and the address of the module in the properties window of the unit connector:

If you are accessing SIMATIC S7-300 and S7-400, enter LADDR in the format "16#XXXX" as the address. The LADDR must start with 16#. This is followed by a four-digit hexadecimal number, with leading zeros if required. The hexadecimal number can only contain digits and uppercase letters, but no lowercase letters.

If you access SIMATIC S7-1500, enter the HWID as a decimal number without leading zeros as the address.

Connect the unit connector with the "Unit" input of the component.



Access to a data record in the distributed I/O is possible with both PROFIBUS and PROFINET.

See also

ReadMemory – Reading a bit memory address area (Page 575)

WriteMemory – Writing to a bit memory address area (Page 575)

ReadDatablock – Reading a data block (Page 576)

WriteDatablock – Writing to a data block (Page 577)

Unit connector (Page 437)

Data record communication (Page 137)

ReadRecordHandler – Response to a read request of the controller (Page 578)

WriteRecordHandler – Response to a write request from the controller (Page 578)

2.14.1.8 Converting the data width of signals

Introduction

The signals that belong to an imported configuration are automatically created in the coupling when system blocks are imported. The system differentiates between binary signals with the data width "Bit" and signals with the data width "Byte", Word (2 bytes), Double Word (4 bytes) and others.

You can change the data width, for example by combining eight consecutive binary signals in one byte signal.

You can only convert signals that have been created in the coupling as a result of the import of system blocks. The address areas are set by the configuration. They cannot be changed, not even by converting the signal data width.

Converting signals

The table below shows how you can convert signals:

- Combining signals: Read a table row from left to right
- Splitting a signal: Read a table row from right to left

Number of signals	Data type	Data width		Number of signals	Data type	Data width
8	BOOL	1 bit	↔	1	BYTE	1 byte
2	BYTE	1 byte	↔	1	WORD (INT)	2 bytes
2	WORD (INT)	2 bytes	↔	1	DWORD (DINT, REAL)	4 bytes

See also

Splitting a signal (Page 234)

Combining signals (Page 235)

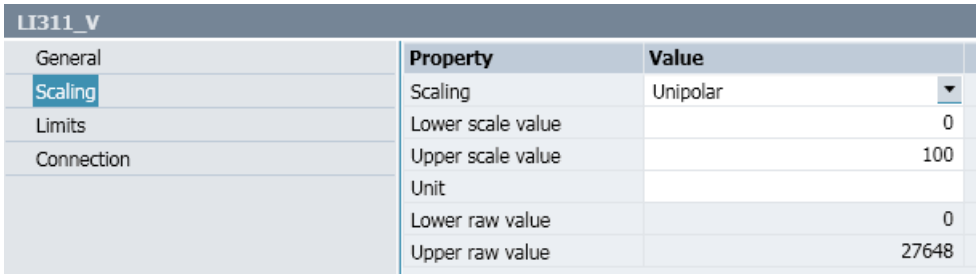
2.14.1.9 Scaling analog signals

In SIMATIC systems and generally also in other automation systems, the values of analog signals are converted into a fixed integer format. The value range is derived from the resolution of the A/D converter and is typically within the range –27648 to +27648 for the SIMATIC S7.

Analog values are treated like floating point values in the simulation. The simulation generally uses the absolute values of physical variables such as pressure, temperature, etc. As input values for the connected automation systems, these values must not only be converted into fixed decimal point format, they must also be mapped onto the range of values for the corresponding measurement ranges. The analog values must therefore be scaled when they are sent from SIMIT to SIMATIC, and rescaled when sent in the opposite direction.

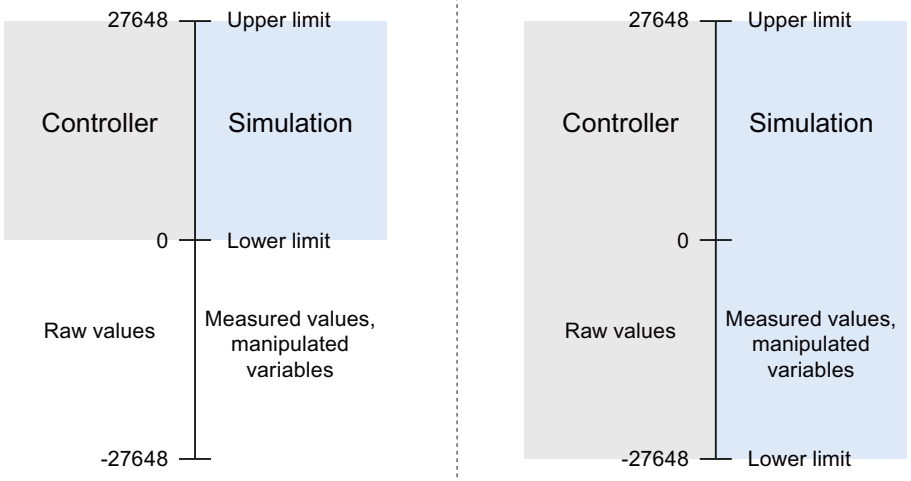
The scaling of input signals and the rescaling of input and output signals is a way of adapting the simulation to the characteristics of the controller. Scaling and rescaling are therefore performed at the interface between the simulation model and the controller in the couplings.

You can define the scaling for a signal either through its properties or using the signal table.



As all analog modules in the SIMATIC module range that are transferred as a word operate with a resolution of 2 bytes, scaling and descaling in SIMIT is only available for analog values of the data type *WORD*.

The figure below shows scaling for unipolar and bipolar measurements:



- Unipolar measurements only provide positive raw values (0 to 27648)
- Bipolar measurements provide both positive and negative raw values (–27648 to +27648).

The following table lists the scaling/descaling supported by SIMIT:

Scaling type		**Measuring range**		**Raw values**	
Number	**Name**	**Start**	**End**	**Lower**	**Upper**
0	No scaling				
1	Unipolar	0 (Default)	100	0	27648
2	Bipolar	–100 (default)	100 (Default)	–27648	27648
3	User-defined	0 (Default)	100 (Default)	0 (Default)	27648 (Default)
4	PT x00 standard	–200 °C	850 °C	–2000	8500
5	PT x00 climate	–120 °C	130 °C	–12000	13000
6	Ni x00 standard	–60 °C	250 °C	–600	2500
7	Ni x00 climate	–60 °C	250 °C	–6000	25000
8	Cu 10 standard	–200 °C	260 °C	–2000	2600
9	Cu 10 climate	–50 °C	150 °C	–5000	15000
10	Type B thermocouple	0 °C	1820 °C	0	18200
11	Type E thermocouple	–270 °C	1000 °C	–2700	10000
12	Type J thermocouple	–210 °C	1200 °C	–2100	12000
13	Type K thermocouple	–270 °C	1372 °C	–2700	13720
14	Type L thermocouple	–200 °C	900 °C	–2000	9000
15	Type N thermocouple	–270 °C	1300 °C	–2700	13000
16	Type R, S thermocouple	–50 °C	1769 °C	–500	17690
17	Type T thermocouple	–270 °C	400 °C	–2700	4000
18	Type U thermocouple	–200 °C	600 °C	–2000	6000

For types 1 and 2, the "Lower scale value" and the "Upper scale value" are preset with 0 or –100 and 100. These values can, however, be adapted to the measurement or setting ranges. Unipolar and bipolar scaling (types 1 and 2) and the "User-defined" type (type 3) can also be used as descaling for output signals.
Scaling types 4 to 18 can only be used as scaling for temperature measurements for input signals.

Note

For the temperature measurement signals in a coupling, the measuring range limits are only displayed. They cannot be edited.

High and low raw values can only be edited if the scaling type "User-defined" has been selected.

Changing the scaling while the simulation is running

When the simulation is running, the measuring ranges of the scaling and the raw value ranges can be changed (for the "Custom" type). Changes take effect immediately.

Note

This function is not available with the Virtual Controller coupling.

Note

The changes only apply to the simulation in progress. They are not automatically applied to the configuration.

Transfer of floating point values (Float)

Some I/O devices - including those in the PROFIBUS PA range - transfer their measured values directly to the controller as physical variables in floating point format with a data width of 4 bytes (double word). In order to handle these floating point values correctly, the corresponding signals have to be assigned the data type *REAL* in the coupling.

See also

Signal table (Page 223)

Scaling signals (Page 238)

2.14.2 Importing and exporting signals

2.14.2.1 File formats for signals

You can switch signal configurations between couplings by exporting and importing signal properties. The signals of a coupling are exported and imported in specific formats. The following file formats are supported in line with the coupling type:

Coupling type	Signal tables	
	Import format[1]	Export format
SIMIT Unit	asc, seq, txt or xlsx	txt, seq
Virtual controller[2]	asc, seq, txt or xlsx	txt, seq
PLCSIM	asc, seq, txt or xlsx	txt, seq
PLCSIM Advanced	asc, seq, txt or xlsx	txt, seq
PRODAVE	asc, seq, txt or xlsx	txt, seq
OPC DA server / OPC UA server	ini, txt	txt, ini
OPC DA / UA client	ini, txt	txt, ini
gPROMS	Init, txt, spreadsheet(xlsx) [3]	Init,txt, spreadsheet(xlsx) [3], json[4]

[1] Only files that are encoded as SBCS (Single Byte Character Set) and use the permanently set code page 1252 ("Western European") are supported.

[2] Only the format "*.txt" is supported for the import and export of data block tables.

[3] This format was defined specifically for use within "Joint Solution" projects; for universal import and export, the "*.ini" and "*.txt" formats are recommended.

[4] This export format is used specifically for support during configuration of the gPROMS gFPI; for universal import and export, the "*.ini" and "*.txt" formats are recommended.

Additional information on signal import and export:

- The standard format is txt format.
- A signal table in asc format or seq format corresponds to the symbol table of a SIMATIC project.
  You can find additional information on this in section: Symbol table (Page 222).
- Exported signal tables contain all the signals of a coupling and their properties such as name (symbol), address, comment, etc.
  You can find additional information on this in section: Signal table (Page 223).

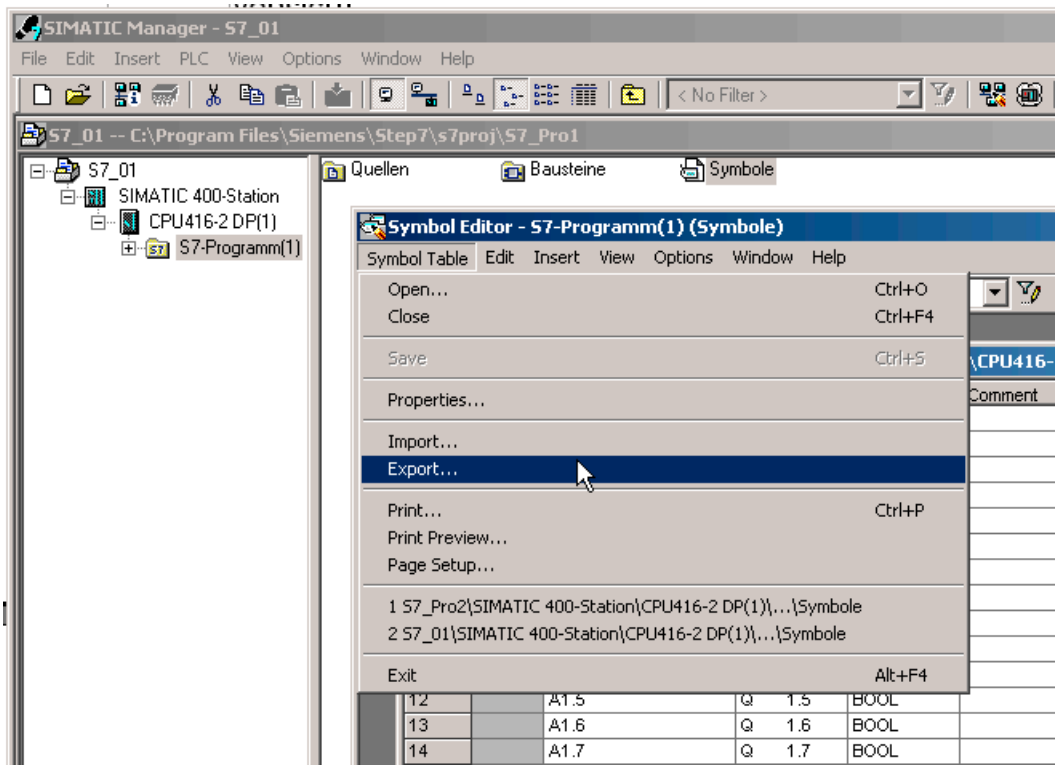
2.14.2.2 Symbol table

The symbol table is exported from SIMATIC Manager from a SIMATIC project in various formats. For import to SIMIT, select asc or seq format.

Note

Please note that in seq format, the comment is limited to 40 characters and no data types are included.

When seq format is imported to SIMIT, data words are always assigned the data type *WORD* and double words the data type *DWORD*. Changes to data types *INT* or *REAL* must then be made manually.



The import of the symbol table is supported by all SIMATIC couplings in SIMIT.

Note

The asc format uses fixed column widths. If you edit this file with a text editor other than the symbol editor, you must not change the number of characters per line.

2.14.2.3 Variable table

The PLC variable lists (symbol tables) are imported from the TIA Portal. The TIA Portal export these lists in xlsx format. Only input and output signals with following data types are transferred:

TIA Portal	STEP 7	SIMIT
Bool	BOOL	Binary
Bytes	BYTE	integer
Word	WORD	integer
Int	INT	integer
DWord	DWORD	integer
Dint	DINT	integer
Real	REAL	analog

2.14.2.4 Signal table

The signal table is structured as follows:

Column	Title	Heading	Meaning
1	Symbol	Symbol	Symbolic name of the signal
2	I/O	InOut	Encoded as *I, Q, IB, QB, IW, QW, ID, QD*, or in the corresponding international notation (I/Q)
3	Address	Address	Absolute address of the signal, e.g. *0.0* or *512*
4	Type	Type	Signal data type: *BOOL, BYTE, WORD* or *DWORD*
5	Comment	Comment	Text as comment
6	Default	Default	Initialization value that is the default of this signal
7	Signal source	ImplicitSource	Source of implicitly interconnected signal
8	Signal name	ImplicitSignal	Signal source of implicitly interconnected signal
9	Unit	Unit	Unit of the signal, e.g. physical unit for measured values
10	Scaling	Scaling	Type number of the scaling (for information, see the table in the section: Scaling analog signals (Page 218)
11	Low limit	ScalingLowerPhys	Physical low or high limit of analog signals:
12	High limit	ScalingUpperPhys	
13	Low raw value	ScalingLowerRAW	Low or high raw value of scaled analog signals. Can only be freely selected with "Custom" scaling type.
14	High raw value	ScalingUpperRAW	
15	Limiting on/off	LimitActive	Limitation of an analog signal is in enabled / disabled
16	Low limit	LimitLowerPhys	Low physical value of the limit
17	High limit	LimitUpperPhys	High physical value of the limit
18	Multiplier	Multiplier	OPC client, cycle multiplier
19	Can be read back	Readback	OPC client, signal can be read back

In txt format, the tabulator is used as the column separator.

The first row must contain the identifier "#Signal properties;". This row contains information on the file type. The SIMIT version number with which this table was exported and the coupling name are also added during export from SIMIT.

The selected properties appear in the second row as column headers. These are displayed even if no corresponding content is available.

One signal and its properties are described in each of the subsequent rows.

Note

The signal table has been revised with the SIMIT version 8.1 and expanded with column headings. If you import a signal table that was created with a previous version of SIMIT, you will need to modify it accordingly.

Due to the new "Custom" scaling type, the significance of the type numbers for scaling has moved.

Note

If you edit a signal table in EXCEL, all columns must be formatted as "text" to ensure that EXCEL does not carry out any unwanted format conversions.

2.14.2.5 INI format of the OPC couplings

OPC signals have their own import/export format for the following reasons:

- They have no addressing function.
- The name does not indicate the direction of transfer.
- There is no conversion of raw values to physical values or vice versa.

The signal table for OPC signals has the INI format. The format consists of 6 sections that are identified with the following keywords:

Keyword	Meaning
[AIN]	Analog input signals
[IIN]	Integer input signals
[BIN]	Binary input signals
[AOUT]	Analog output signals
[IOUT]	Integer output signals
[BOUT]	Binary output signals

Each keyword must be in a separate row. This is followed by the signal names to be assigned to this section. The order of sectors is arbitrary. Not all sections must be used.

In each section, the signal table contains a single line per signal. The signals and their properties are exported in quotation marks and separated by a semicolon.

	OPC UA server	OPC DA server	OPC client
Inputs	• Name • Comment • Default • Implicit interconnection (source/target) • Implicit interconnection (signal) • Can be read back ("True/False")	• Name • Comment • Default • Implicit interconnection (source/target) • Implicit interconnection (signal)	• Name • Comment • Default • Implicit interconnection (source/target) • Implicit interconnection (signal) • Multiplier • Can be read back ("True/False")
Outputs	• Name • Comment • Implicit interconnection (source/target) • Implicit interconnection (signal)	• Name • Comment • Implicit interconnection (source/target) • Implicit interconnection (signal)	• Name • Comment • Implicit interconnection (source/target) • Implicit interconnection (signal) • Multiplier

If a row contains only a signal name without quotation marks when the signal table is imported, the signal will be imported with that name, without comments and with the defaults.

Note

In the case of the OPC client, the import format of SIMIT V8.1 is not backwards compatible with older versions of SIMIT. If you have specified the multiplier or the readback capability in existing import files, you must now add 2 additional empty columns for the implicit interconnection.

2.14.2.6 Spreadsheet format (xlsx) of the gPROMS coupling

The gPROMS coupling supports the configuration of the interface via import and export of spreadsheet documents.

This format is specifically tailored to requirements for use within the framework of a co-simulation with PSE/gPROMS® process models.

The Excel file format is thereby supported (.xlsx). The export of the configuration generates an Excel spreadsheet with the following structure.

Sheet column heading	Meaning	Permitted values
Symbol	OPC UA tag name	
Direction	Direction of signal (SIMIT)	Input Output

Type	Data type of signal (SIMIT)	Analog Integer Binary
Comment	Description of signal	Free text
Default	Value upon initialization	Dependent on data type
ConnectionSource	Block/coupling name of interconnection destination	
ConnectionSignal	Signal/connection name of interconnection destination	
ReadBack		True / False
gPROMSModelName	Name of signal for use within the gPROMS modeling	
gPROMSClientTag	Tag name onto which this signal is mapped for use within the PSE/gPROMS® modeling	
RCICmdType	Assignment of the signal to a command specified in RCI. Only the commands needed for the co-simulation with PSE/gPROMS® process models are taken into consideration here.	RCINoCmd RCIRun RCIStopRCIClose RCICreateSnapshot RCILoadSnapshot gFPISpeedUp

Note

Note the following: The sheets used by this coupling type are not identical and not compatible with the symbol tables that can be exported from the TIA Portal.

2.14.2.7 JSON format of the gPROMS coupling

The signal interface that is being used by PSE/gPROMS® process models within the gFPI is based on the OPC UA protocol.

The data structures needed for the configuration are written within the PSE/gPROMS® modeling in JSON format.

Definition	Label	Properties	Meaning

HEADER	{
	"opcServer":
		{
		"modelTimeUoM":
		"namespace":
		"realTimeSpeedUp":
		"serverName":
		}
			Unit of time: "s", "m", "h"
			Namespace of tag
			Factor
			URI of OPC UA server

BODY	"opcItems":
	[
	
	]
	}
		{
		"clientTag":
		"description":
		"serverTag":
		}
			Name of signal in model
			Comment
			Name within the namespace

		{
		"clientTag":
		"description":
		"serverTag":
		}
			Name of signal in model
			Comment
			Name within the namespace

The format presented above contains no information about the data type and signal direction.

The context of the PSE/gPROMS process model yields this information within the gFPI.

Note

Due to the above-indicated limitations of this format, the gPROMS coupling only offers JSON format for exporting.

2.14.2.8 Importing signal properties

Introduction

You can import signal properties from the following files:

- Symbol table
- Variable table
- Signal table
- INI file

Requirement

- Project is open.
- Import file is available.
- Coupling has been created.
- Coupling editor is open.

Procedure

Follow these steps to import signal properties:

1. In the coupling editor, open the dialog box for importing signal properties.
2. Select the format of the import file.
3. Select the import file.
4. Select the signal properties you want to import.
5. Over "Mode", select what is to happen to the properties of existing signals.
6. If required, select "Adapt data width".
7. Click "Import".

Result

The properties are imported.

See also

"Importing signal properties" dialog box (Page 988)

2.14.2.9 Importing PLC variable lists

PLC variable lists (symbol tables) are imported from the TIA Portal. These lists are exported from the TIA Portal in EXCEL format (*.xlsx). Only I/O signals with the data types listed in the table below are copied from the variable list:

Table 2-6 Comparison of data types

TIA Portal	STEP 7	SIMIT
Bool	BOOL	binary
Bytes	BYTE	integer
Word	WORD	integer
Int	INT	integer
DWord	DWORD	integer
DInt	DINT	integer
Real	REAL	analog

2.14.2.10 Exporting signal properties

Requirement

- Project is open.
- Coupling has been created.
- Coupling editor is open.

Procedure

Follow these steps to export signal properties:

1. In the coupling editor, open the dialog box for exporting signal properties.
2. Select the format of the export file.
3. Select the storage location of the export file.
4. Select the signal properties you want to export.
5. Click "Export".

Result

The properties are exported.

See also

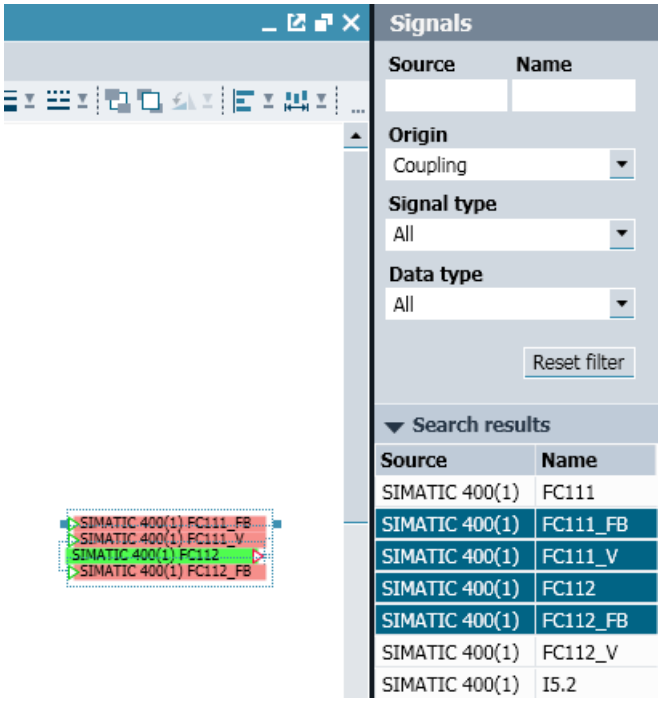
"Exporting signal properties" dialog box (Page 991)

2.14.3 Using I/O signals

2.14.3.1 I/O signals for connection on charts

I/O signals can be used in charts for application to the simulation model.
Proceed as follows:

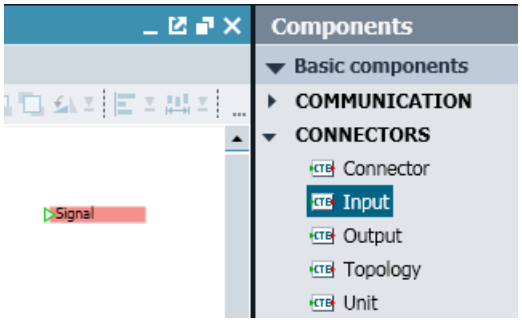
1. Open the required chart in the chart editor.
2. Open the "Signals" task card.
   Set the "Origin" filter to "Coupling" to display I/O signals only.
   You can find additional information on the task card in: "Signals" task card (Page 270).
3. Select one or more signals from the task card and drag-and-drop it/them to the chart.
   Controls for signal operation are then generated in the chart.
   If you instead want to generate I/O connectors for the signals, hold down <Shift> when you drag-and-drop. This generates I/O connectors that already have the correct signal names.



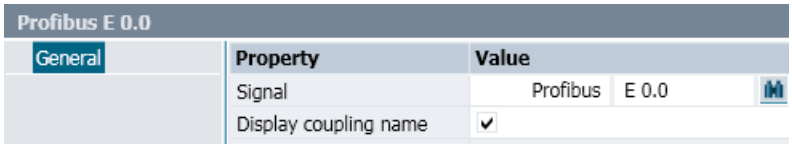
You can now connect the I/O connectors to a connector of another component as desired.
If multiple input I/O connectors or multiple output I/O connectors are selected in a multi-selection, you can use the "Adapt selection" shortcut menu to adapt all connectors of this selection to the length of the connector selected as primary.

Dragging I/O signals from the "Components" task card

I/O connectors from the "Components" task card can also be used; in this case, however, the coupling and signals names must be entered manually.



To enter the signal name, simply click on the I/O connector to select it and enter the name of the coupling and the signal name in the property view.

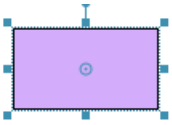


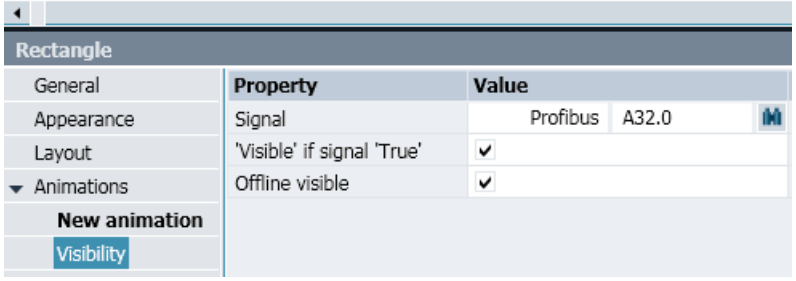
Note

The name of the I/O connector must be written exactly as it is entered in the configurator. Therefore, pay attention to upper/lower case and do not use spaces in address names such as *I0.0*.

2.14.3.2 I/O signals for animations

You can use I/O signals to animate graphic objects. To do this, enter the signal name in the property view of the animation.

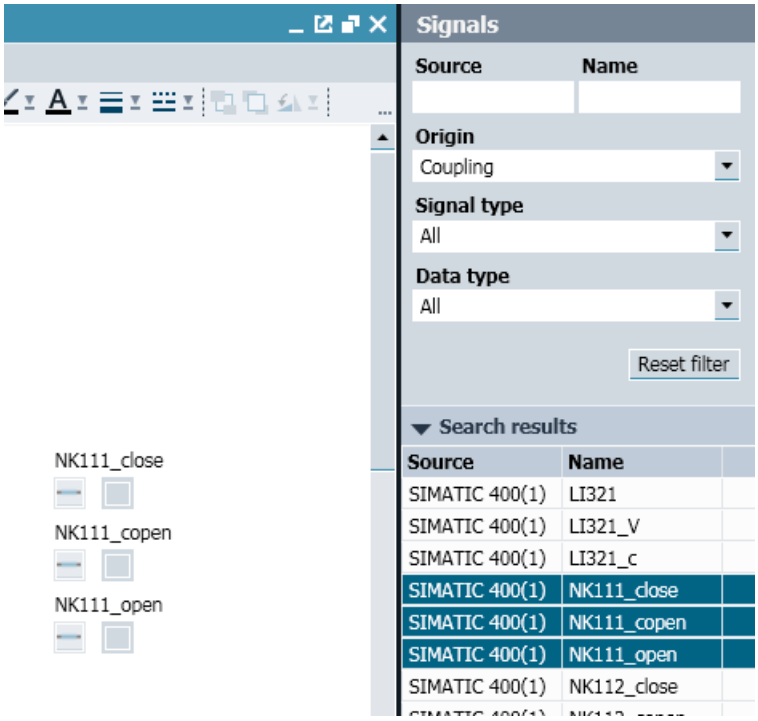




You can drag-and-drop the signals from the "Signals" task card.

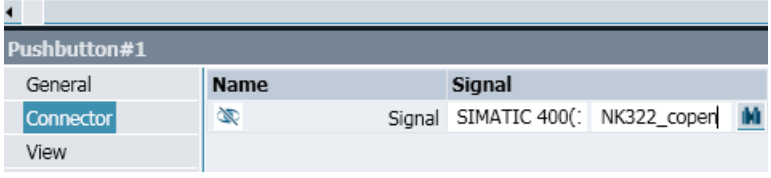
2.14.3.3 I/O signals in controls on charts

The controls with fixation can be copied from the coupling into the chart. To do this, drag the I/O signals from the "Signals" task card into the chart. The name for the control is copied as a text element to the chart.



You can also use I/O signals directly in other controls, such as a pushbutton.





Enter the name of the I/O in the property screen of the control or drag-and-drop the I/O signal from the "Signals" task card to the property view of the control.

2.14.3.4 I/O signals in trends

You can also use I/O signals in trends. Follow these steps:

- Enter the signal with source and name in the property view of the trend.
- Drag-and-drop the signal from the "Signals" task card to the trend property view.

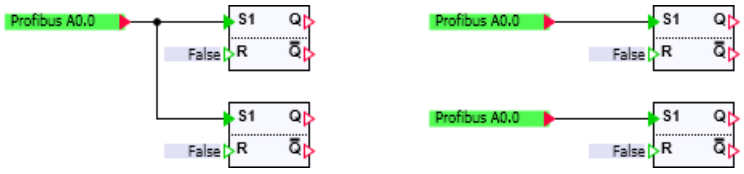
Properties

	Source	Name	Color	Range	Cursor	Alias
	SIMATIC 400 (1)	NK111_close		Binary		
	SIMATIC 400 (1)	NK112_open		Auto (0 .. 0)		
*						

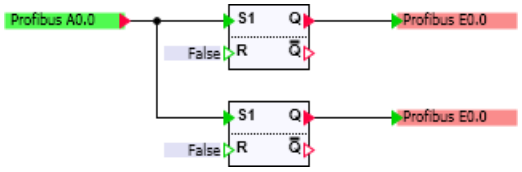
You can find more information on trends in: Trends (Page 314).

2.14.3.5 Multiple use of I/O connectors

I/O input connectors with the same name can be used multiple times in chart. Each of the two figures below is therefore possible:



An I/O output connector can only be used once in an entire project. The following interconnection would result in an error when the simulation project is started:



2.14.4 Interconnecting signals

Introduction

An interconnection is the exchange of signals between two objects:

- Two coupling signals
- One coupling signal and the input or output of a component



Requirement

- Coupling has been created.
- Coupling editor is open.

- Signals have been created.
- Signals to be interconnected are of the same data type.
- The signal is not interconnected.

Procedure

Follow these steps to interconnect signals:

1. Select the signal.
2. Enter the connection destination under "Connection" in the signal properties.

Result

The signal is interconnected.

2.14.5 Splitting a signal

Introduction

If you split a signal, its symbolic name is removed.

Requirement

- Project is open.
- One of the following coupling types has been created:
  - SIMIT Unit
  - Virtual controller
  - PLCSIM Advanced
- Coupling editor is open.
- The signal to be split:
  - Has been imported from a system block
  - Is not fail-safe

Procedure

Follow these steps to split a signal:

1. Select a signal with one of the following data types from the signal table:
   - BOOL
   - WORD
   - DWORD
2. Select "Split" from the shortcut menu.

Result

The signal is split into up to eight signals in line with the data type.

See also

Converting the data width of signals (Page 217)

2.14.6 Combining signals

Introduction

When you combine signals, their symbolic names are removed.

Requirement

- Project is open.
- One of the following coupling types has been created:
  - SIMIT Unit
  - Virtual controller
  - PLCSIM Advanced
- Coupling editor is open.
- Signals to be combined:
  - Have been imported from system blocks
  - Are located in the same module
  - Are not fail-safe

Procedure

Follow these steps to combine signals:

1. Select consecutive signals from the signal table:
   - Data type BOOL: 8 signals
   - Data type BYTE or WORD: 2 signals
2. Select "Combine" from the shortcut menu.

Result

The signals are combined.

See also

Converting the data width of signals (Page 217)

2.14.7 Preassigning signals

Requirement

- Project is open.
- One of the following coupling types has been created:
  - SIMIT Unit
  - Virtual controller
  - PLCSIM
  - PLCSIM Advanced
  - PRODAVE

Procedure

Follow these steps to preassign a value to a signal:

1. Open the coupling editor.
2. Allocate the signal under "Inputs > Default":
   - Analog signal: Enter the required value.
   - Binary signal: Select the check box.

Result

The signal is preassigned the specified value when simulation is launched. The value can change during the course of simulation.

See also

Fixing a signal (Page 237)

2.14.8 Fixing a signal

Introduction

In the coupling editor, you can set signals and display signal values while simulation is running.

Requirement

- Coupling has been created.
- Simulation has started.
- Coupling editor is open.

Procedure

Follow these steps to fix a signal:

1. Either enter a value or select the check box in line with the data type of the signal.
2. Select the fixing switch.

Result

The value of the signal is fixed. The value is not changed during simulation.

See also

Fixing signals in the coupling editor (Page 212)

Preassigning signals (Page 236)

2.14.9 Addressing a signal symbolically

Introduction

In SIMIT, signals are by default uniquely identified with their address. As the address says little about the function of a signal, you can with some couplings assign a descriptive symbolic name for a signal. Addresses or symbolic names are, for example, displayed at input and output connectors.

Requirement

- Project is open.
- One of the following coupling types has been created:
  - SIMIT Unit
  - Virtual controller
  - PLCSIM
  - PLCSIM Advanced
  - PRODAVE

Procedure

Follow these steps to address a signal symbolically:

1. Open the coupling editor.
2. Under "Symbol name", enter a symbolic name for the signal.
3. Select the "Options > Assign coupling signals" menu command.

Result

The identification of the signal has been switched from absolute addressing to symbolic addressing. In SIMIT, you can now only access the signal using its symbolic name.

See also

Options > Assign coupling signals (Page 981)

2.14.10 Scaling signals

Introduction

You can enter scaling values in the signal table instead of in the properties of a signal.

Requirement

- Project is open.
- One of the following coupling types has been created:
  - SIMIT Unit
  - Virtual controller
  - PLCSIM
  - PLCSIM Advanced
  - PRODAVE
- Signal is analog.
- Signal is of the data type WORD.

Procedure

Follow these steps to scale a signal:

1. Open the coupling editor.
2. Select the signal.
3. Configure the scaling values in the signal properties under "Scaling":
   - Select the scaling type.
   - Enter the lower and upper physical value.

Result

Scaling for the signal has been defined.

See also

Signal table (Page 223)

Scaling analog signals (Page 218)

Transferring scaling to another signal (Page 239)

2.14.11 Transferring scaling to another signal

Introduction

You can transfer scaling from one signal to one or more other signals. Scaling is transferred in the coupling editor column by column.

Requirement

- Project is open.
- One of the following coupling types has been created:
  - SIMIT Unit
  - Virtual controller
  - PLCSIM
  - PLCSIM Advanced
  - PRODAVE
- Coupling editor is open.
- Signal has been scaled.
- A scaling type is set at the destination signal.

Procedure

Follow these steps to transfer the scaling to another signal:

1. Select the signal from the "Scaling" column.
2. Select "Copy cell" from the shortcut menu.
3. Select the destination signal from the "Scaling" column.
4. Select "Paste to cell" from the shortcut menu.

Result

The scaling is transferred to the destination signal:

- Scaling type
- Lower and upper physical value

See also

Scaling signals (Page 238)

2.14.12 Limiting a signal

Introduction

You can define a limit range for all analog coupling signals. Values outside this range many not be transferred to the coupling partner or the simulation depending on the signal:

- Limit range for input signal: Coupling partner
- Limit range for output signal: Simulation

Requirement

- Project is open.
- One of the following coupling types has been created:
  - SIMIT Unit
  - Virtual controller
  - PLCSIM
  - PLCSIM Advanced
  - PRODAVE
- Signal is of the SIMIT data type "analog":
  - Standardized measured value (WORD)
  - Floating-point number (REAL)

Procedure

Follow these steps to limit a signal:

1. Open the coupling editor.
2. Select the signal.
3. Configure the limits in the signal properties under "Limits":
   - Activate the limits.
   - Enter the lower and upper physical value.

Result

A limit range is defined for the signal. You can disable or change the limits while simulation is running.

2.14.13 Reading signals from an OPC server

Note

When you read OPC signals, all signals in the coupling editor are overwritten. If you have input cycle multiplier or comments at signals, for example, export the signals before reading. Once you have read the signals from the OPC server, import the signals again.

Requirement

- Project is open.
- One of the following coupling types has been created:
  - OPC DA client
  - OPC UA client
- Connection to the OPC server is configured.
- OPC server can be accessed in the network.

Procedure

Follow these steps to read signals from an OPC server:

1. Open the coupling editor for the OPC client coupling.
2. Import the OPC signals with "Browse".

Result

The OPC signals are read from the OPC server. Only OPC signals whose data types are supported by SIMIT are entered. Whether a signal is created as "Input" or "Output" depends on the configuration of the signal on the OPC server.

Only nodes with the 'Variable' NodeClass, the 'String' IdentifierType and the selected namespace are taken into account when browsing the tag management of an OPC UA server.

See also

Mapping of signals and data types of an OPC server (Page 168)

Configuring an OPC DA client coupling (Page 173)

Configuring an OPC UA client coupling (Page 182)

2.14.14 Configuring a signal "With readback capability"

Introduction

Signals transferred from SIMIT to an OPC server need not necessarily be accepted and applied by the OPC server. To make the value that is actually effective in the OPC server available in the simulation model, configure the signal as "can be read back".

Likewise, signals that SIMIT provides as an OPC UA server may not be read by the OPC UA client during the normal I/O cycle.

In order to have the last value that was actually in effect in communication partners available in the simulation model, configure the signal as "can be read back".

Requirement

- Project is open.
- One of the following coupling types has been created:
  - OPC DA client
  - OPC UA client
  - OPC UA server
  - gPROMS
- Coupling editor is open.
- Signal is configured as "ReadWriteable" at the OPC server.
- Signal has been crated under "Inputs" in the coupling editor.

Procedure

Follow these steps to configure a signal as "With readback capability":

1. Select the signal.
2. In the properties of the signal, select the "Signal with readback capability" option.

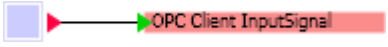
Result

The signal is configured as "can be read back". You can use the input signal in an output connector.

If an OPC client coupling is involved, the value that is actually in effect on the OPC server can be acquired via this connector.

This connector can be used within an OPC UA server coupling to evaluate the value that was last read by the OPC UA client.

- Value that is sent to the OPC server by SIMIT:



- Value that is read back from the OPC server:



2.15 TableReader coupling

2.15.1 Functionality of the TableReader coupling

2.15.1.1 Functionality of the "TableReader" coupling

With the "Table Reader" coupling, SIMIT reads fixed values from a table. The values read from the table are available as input values in SIMIT and are connected to input signals. SIMIT reads a single row of the table, which is selected by an index. You switch between different parameter sets using the index.

2.15.1.2 Structure of a TableReader table

During configuration, the first row of the table is evaluated. Each column defines a signal name for the coupling. Default values for these signals are expected in the second row. Integers, fixed-point numbers and binary values are supported.

An "Index" input signal is also created, which can be used to select the row to be read in while simulation is running. Index 0 addresses the row of the presets, index 1 the next row and so on.

	A	B
1	On/Off	Speed
2	0	0
3	1	1500
4	1	3000

If a read entry in the table does not correspond to the specified number format, the signal value is set to 0 or false.

If the index is higher than the existing rows in the table, all values are set to 0 or false.

2.15.2 Configuring the TableReader coupling

2.15.2.1 Creating a TableReader coupling

To create a "Table Reader" coupling, select the "TableReader" option in the "Selection" dialog box. A maximum of 16 instances of this coupling can be created in a project.

2.15.2.2 Importing a table

To import a table into Table Reader, select the "Configuration" menu item and select the table in the file directory.

The following file formats are allowed:

- .xlsx
- .txt
- .csv

Note

The table is not copied to SIMIT when the coupling is configured. Values in the table can still be edited after the table has been imported as long as no simulation is running. When archiving the project, the table must be archived separately.

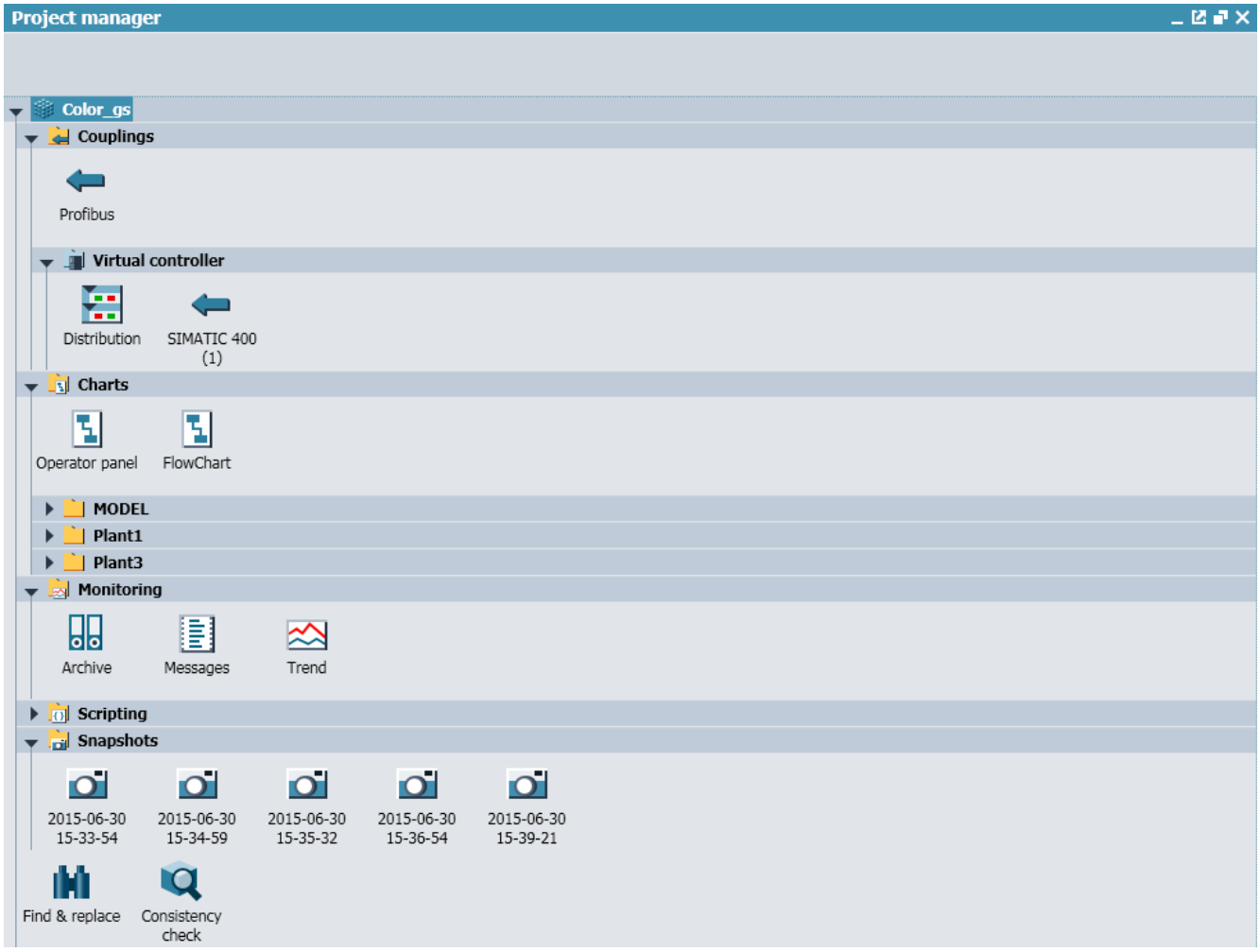
Simulation model

3

3.1 Project manager

3.1.1 View and functions of the Project Manager

You open the project manager with a double-click on the tree entry Project manager in the project navigation.
An alternative view of the project is displayed in the work area for project navigation:



The following editing options are available in the project manager:

- Moving charts, components of the project and entire folders using drag-and-drop
- Copying, pasting, deleting and renaming elements using shortcut menu commands
- Opening elements of the project such as charts for editing with a double-click

See also

Versioning (Page 249)

Write protection (Page 251)

Operating modes (Page 39)

Scalability (Page 878)

3.1.2 Properties

The following properties of your SIMIT project are displayed in the property view and can be edited:

- **General**
  - Project location
    The folder in which you have stored the project. This property cannot be modified here.
  - Project version
    You can find more information on this in section: Versioning (Page 249).
  - Read-only
    You can find more information on this in section: Write protection (Page 251)
  - "Default scale", if you are using the CONTEC library.
    You can find more information on this in section: Scalability (Page 878).
- **Times & operating modes**
  - 8 different time slices (Time slice 1 to Time slice 8)
    You can assign the 8 time slices to the components, controls and couplings in your project. For the 8 time slices, you can specify any value in milliseconds, provided it is at least 1 ms.
  - Operating modes
    You can find more information on this in section: Operating modes (Page 39).

- **Backtracking**
  - Active
    Activates the "Backtracking" function.
    By means of the "Backtracking" function, the states (backtracks) of a simulation are saved at regular intervals.
    You can find more information in "Available backtracks" dialog box (Page 983)
  - Interval
    Defines the time intervals in which the backtracking runs.
    The minimum interval is 30 seconds.
  - Maximum number of backtracks
    Between 10 and 1000.

Note

If less than 5 GB of memory space is available on the hard disk, no backtrack will be stored. An indication of this is entered.

The "Backtracking" function is not available in isochronous mode.

If backtracks are saved during a simulation with a virtual controller, no changes to the S7 program are permitted to be downloaded to the virtual controller during the simulation. Otherwise, the saved backtracks are no longer usable. The saved backtracks are deleted when simulation is exited.

- **Engineering**
  - Server
    Select an engineering server that is connected to SIMIT.
  - Project
    From the engineering server, select an engineering project to be imported in SIMIT.

Note

The "Server" and "Project" options can also be selected when creating a project.

You will find more information in Project > New project ... (Page 974).

When an engineering project is selected, the hardware configuration of the selected engineering project can be imported directly using a virtual controller import.

You will find more information in "Virtual controller import" dialog box (Page 985).

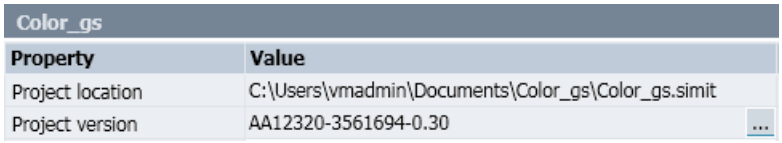
3.1.3 Versioning

The version number is automatically included in the simulation project. It consists of several elements:

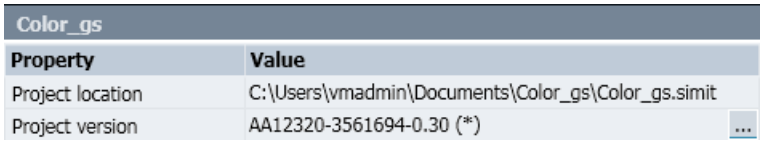
- License number (e.g. AB12345)
- Time stamp (resolution is approx. 38 seconds)

- Major version (0 - 127)
- Minor version (0 - 999)

If the simulation project was changed, the version number is updated automatically the next time the simulation is started.



If the simulation project has been changed but has not been started since then, an asterisk appears after the version number.



The minor version is incremented to a maximum of 999; it then jumps back to 0 and the major version is incremented by 1. You can also increment the major version manually by clicking the "..." button. When you increment manually, a message appears that you need to confirm. The project is then compiled and the new version identifier entered.

Note

If you create a new simulation project or open or retrieve a simulation project created with a previous version of SIMIT, the version is set to "unversioned". The version is set automatically once the simulation is started.

Viewing the versioning with the "ProjectVersion" component type

Another way to display the version identifier is with the "ProjectVersion" component type. This component type can be found in the "Misc." subfolder of the basic library.

You can find additional information on this component type in: ProjectVersion – Project version (Page 494).

Viewing the versioning with a system variable

When the Automatic Control Interface function module is used, the project version can be accessed via the *_ProjectVersion* system variable. This means that the project version can also be output in the log file: To do so, create a script with following program code:

```
OPEN-LOG "C:\\Logfile.txt"
PRINTF "project version: %s", "_ProjectVersion"
CLOSE-LOG
```

3.1.4 Write protection

You can protect parts of the simulation project or the project as a whole from changes. All elements in a project that can be protected have the additional property "write-protected".

Setting and removing write protection

If you want to use or cancel write protection for a particular folder, this action can be applied to all subfolders and the objects that they contain. A message then appears that you need to confirm.

The write protection can also be changed individually for the objects in subfolders; it does not have to be the same as the folder status.

Proceed as follows to set or cancel write protection for an object:

1. Select the object.
2. Select or deselect the check box for the "Write-protected" property:

The effects of write protection

The effect of write protection varies depending on the objects in a project to which it is applied:

- **Chart**
  Read-only charts cannot be deleted or renamed. They can be opened in the editor but not modified. Objects in the chart (components, controls and graphics) can be selected and copied to other charts. Inputs can be controlled while the simulation is running.
- **Chart folders**
  Read-only folders cannot be deleted or renamed. New charts or subfolders cannot be created in read-only folders, not even by moving or copying charts or folders.
- **Couplings**
  Read-only couplings cannot be deleted or renamed. They can be opened but not changed.
- **Couplings folder**
  No new couplings can be created in the read-only couplings folder.
- **Snapshots**
  Read-only snapshots can be loaded when a simulation is running but not deleted or renamed.
- **Snapshot folders**
  Read-only folders cannot be deleted or renamed. Charts or snapshots cannot be created in read-only folders, not even by moving or copying snapshots or folders.
  If the uppermost snapshot folder in the project tree is read-only, no more snapshots can be created, because they are automatically stored in the uppermost snapshot folder. The corresponding symbol in the toolbar is in this case active.
- **Trend**
  Read-only trends cannot be deleted or renamed. They can be opened but not changed.
- **Archive**
  The write-protected archive can be opened but not changed.
- **Script**
  Read-only scripts cannot be deleted or renamed. They can be opened but not changed.
- **Script folders**
  Read-only folders cannot be deleted or renamed. New scripts or subfolders cannot be created in read-only folders, not even by moving or copying scripts or folders.

- **Lists**
  Read-only lists cannot be deleted or renamed. They can be opened but not modified.
- **Project Manager**
  If the Project manager is write-protected, the project properties cannot be changed. However, all actions relating to project elements are available in both the Project Manager and the project tree, subject to their write protection settings.
- **Complete project**
  Projects that are write-protected at the project level cannot be renamed. The properties of these projects are not editable: Properties (Page 248).
  The following functions are not possible in write-protected projects:
  - Activation and deactivation of couplings (Page 60)
  - Activation and deactivation of charts (Page 258)

Note

Please note that write protection for a project does not automatically mean that all parts of the project are protected from changes. However, you can apply write protection to all objects in a project that you are write-protecting.

Write-protected objects open with a white header line in the work area to indicate write-protected status.

3.2 Chart editor

3.2.1 Creating and editing charts

A chart with predefined or self-defined components and controls is created in the chart editor. Proceed as follows to create a new chart:

1. In the project navigation, double-click on the tree entry "New chart".
   A new chart is created immediately below this tree entry.
2. Give the new chart a name or accept the "Chart" default setting. Other new charts are also assigned the default name "Chart" plus a consecutive number.
3. Double-click on the new chart.
   The chart editor opens in the work area.
4. Drag-and-drop the components and controls for your project to the chart editor.
   - You will find the components for various logical and arithmetic functions, for drives, sensors, connections and communication on the "Components" task card under "Basic components".
   - You will find the objects for entering and displaying values on the "Controls" task card. The "Display" area contains objects for the dynamic display of values from a current simulation.
     The "Input" area contains objects for specifying values in a current simulation.

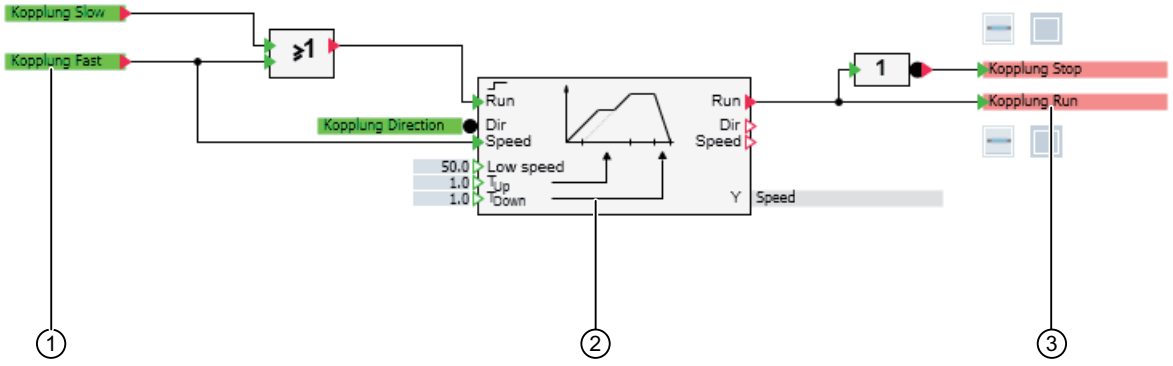
5. Create link views of already placed components with <Ctrl+Shift+Drag & Drop>.
   You can use the link view to open the operating window of the component with a double-click during a current simulation. You can find additional information in section The link symbol (Page 389).
6. If you want to place control elements of component operating windows in the chart:
   – Double-click the operating window of the component to open it.
   – Select the required control elements.
   – Drag the control elements to the chart with drag & drop.

   Consult the component help to determine whether or not a component has an operating window.
7. Connect the I/O of the components and controls to each other and to the controller. The connection to the controller is over connectors.

Note

The minimum size for a diagram is 20 × 20 pixels.

The figure below shows an example of a component with input and output connectors in the chart editor:



① Output connector
② Component
③ Input connector

Connecting I/O

Proceed as follows to connect I/O:

1. Position the mouse pointer over the I/O to be connected.
   If the cursor display changes, a connection is possible at this point.
2. Click on the I/O to be connected.
3. Position the mouse pointer over the connection partner.
4. Click the left mouse button again.
   The connection is now established and is shown by a connecting line.

Alternatively, you can hold down the mouse button to drag a connection from one I/O to another.

Assigning parameters and values to an I/O

You set parameters for a component or a control in the text boxes at the inputs or in the property view of the component or control:

- For input directly at the control/component, open the text box with a double-click.
- For input using the property window, click the component/control to open the property window.

Enter the desired value and confirm the input by pressing the <Enter> key.

Connecting connectors

The input and output signals of the controller are managed in the couplings. These signals are shown on charts by connectors:

- Output signal by green output connectors
- Input signals by red input connectors

You can drag-and-drop input and output connectors from the coupling to charts. Follow these steps:

1. Split the work area with the menu command "Window > Split horizontally".
2. Open the coupling and the chart.
3. Drag the signal of interest from the coupling onto the chart by grabbing the signal at the left border of the coupling and pressing the <Shift> key.
4. Connect the connector I/O to the I/O of a component.

Note

The coupling configuration must first be saved before a signal can be extracted from it.

You can alternatively also drag the input and output connectors from the "Signals" task card to a chart. You can find additional information on this in section: "Signals" task card (Page 270).

3.2.2 Visualizing graphics

Graphics enable you to visualize the simulation model and add an explanatory texts. You can create static and animated graphics and link them together. Graphics are displayed and edited in the chart editor.

Static graphic

To use a static graphic, drag-and-drop the required graphic element from the "Graphics" task card to the chart editor. You can find additional information on this in section: "Graphic" task card (Page 266).

Edit the graphic element with the functions provided in the chart editor toolbar:



The following functions can be called up from the toolbar:

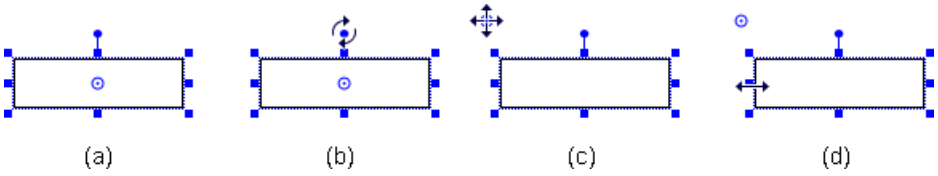
- Text: Set the font, font size and color for text.
- Lines: Select a color for the fill and lines, change the weight and style of lines.
- Graphic element: Flip graphics, mutually align, distribute or group multiple graphics. Group graphics, move individual or grouped graphics to the foreground or background.

Select multiple graphics with a lasso frame or by pressing <Shift> or <Strg>.

Graphics are arranged on a specified grid. You can remove the grid when editing graphics by pressing <Alt>.

You can also change the parameters of graphics in the property view.

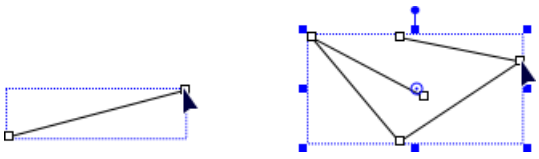
Changing the size and angle of a static graphic



When a graphic object is selected, a selection frame is displayed (as shown in the figure above under (a)). You can change the size and angle of the graphic element using the superimposed blue squares as follows:

- Move the mouse cursor over a square until its appearance changes as shown in the figure above under (d). With the mouse button pressed, drag the graphic element to the desired size.
- You can also rotate the graphic element by any desired angle with the upper blue point. Move the mouse cursor over the blue point until its appearance changes as shown in the figure above under (b). With the mouse button pressed, rotate the graphic element to the required angle.
- You can move the angle of rotation as required (figure above under (c)).

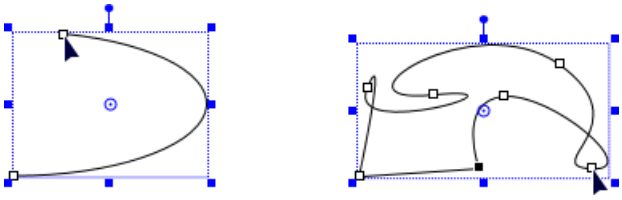
Changing lines



Both endpoints of a line can be moved, whether for a simple straight line or for a line segment within a polyline. Follow these steps:

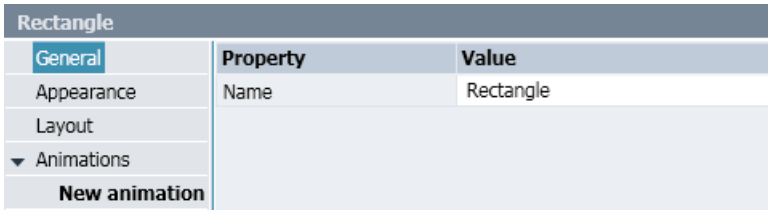
1. Select the line.
   A small rectangle appears at each end of the line.
2. Move the mouse cursor over an endpoint until its appearance changes as shown in the figure above.
3. You can now move the endpoint while pressing the left mouse button.

You can follow the same procedure to move the endpoints of curved lines, in other words the endpoints of elliptical arcs or the points of a Bézier curve:



Animated graphic

Animations can be created for each graphic element. For example, if you draw a rectangle, you can see the "Animations" property in the properties view of the rectangle.



You can create a new animation for the selected graphic element by double-clicking "New animation".

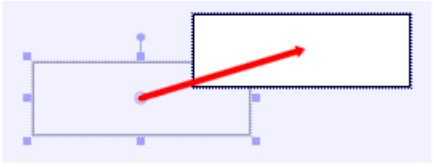
A dialog box opens and you can choose from the following animations:

- **Movement**
  The graphic element moves in the chart.
- **Rotation**
  The graphic element rotates around the axis of rotation.
- **Scaling**
  The graphic element changes size.
- **Visibility**
  The graphic element is shown and hidden.
- **Image change**
  Image files are shown in the graphic element.
- **Image sequence**
  An image sequence of image files is shown in the graphic element.

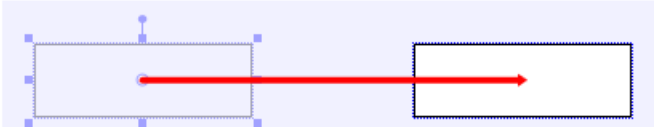
You may define several animations for a single graphic element and can both move and scale an object, for example. Image change and image sequence are, however, mutually exclusive.

Example of "Movement" animation

(a)



(b)

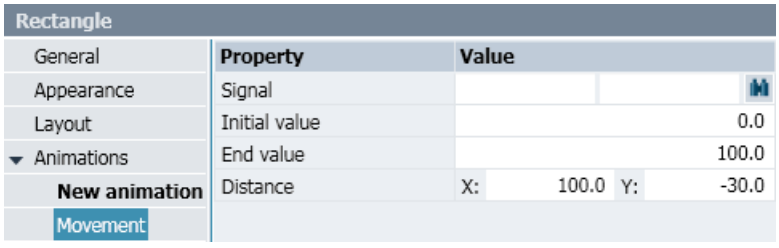


If you select the "Movement" animation, a copy of the graphic element appears offset from the original object. A red arrow connects the copy to the original element as shown in the figure above under (a). This red arrow visualizes the movement of the object.

Use the mouse to move the copy to the required target position.
The figure above under (b) shows the movement and target position for a horizontal movement.

In the properties view (see figure below), enter the signal that supplies the values for the movement.

- To do this, simply drag the signal from the "Signals" task card into the property field.
- Enter the initial value and end value.



If you then start the simulation and change the signal value, the graphic element is moved horizontally.
For signal values within the range specified by the initial and end values, movement takes place along the specified arrow; for signal values outside the defined range, the movement continues in a straight line beyond the arrow. The arrow thus specifies the direction of the linear movement; the initial and end values only define the scale of the graphic element in the chart.

Several movement animations for a graphic object are superimposed and thus allow for movement along curved paths.

For a group of several graphic objects, you can animate either the group as a whole or each individual graphic object within the group.

3.2.3 Visualizing signals

Signals are displayed and set in the charts with controls. The controls can be arranged on a chart in any way, and grouped on charts as required. For each chart, you can specify not just the signals that are displayed but also their arrangement.

To display a signal in the chart editor, drag-and-drop the signal from the "Signals" task card. You can find additional information in section: "Signals" task card (Page 270)
. You can also drag the signal from the coupling editor.

How the signal is displayed depends on the data type:

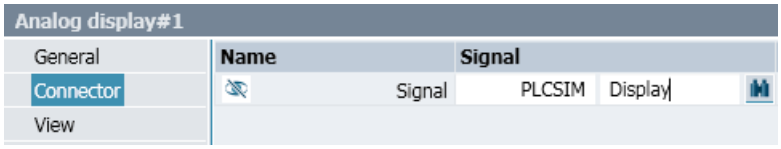
Display of binary signals	Display of analog signals
NK113_open	ADD#4 X2

The current signal name and a signal isolator are also displayed.

A signal can also be displayed simultaneously with several controls, for example, with a digital display and an analog display. You can obtain a digital display by dragging the signal from the coupling or the task card and deleting the signal splitter.
To generate the digital display and the analog display, drag the corresponding control from the "Controls" task card onto the diagram.

You can find additional information in section: "Controls" task card (Page 263).

Either you enter the signal name manually, or it is generated by dragging and dropping the signal from the "Signals" task card to the property view.



3.2.4 Printing charts

The charts and templates can be printed out. The printout has the following characteristics:

- If a folder is selected in the project navigation, all charts in this folder and all subordinate folders are generated as individual print jobs.
- If a chart does not fit onto the selected paper format, it is scaled automatically.
- The size of the chart is indicated by a black frame in the printout.
- Additional information that is printed out:
  - For charts, the chart name including associated folder hierarchy
  - For templates, the absolute file name with path

3.2.5 Activation and deactivation of charts

3.2.5.1 Activation and deactivation of charts

All created charts can be activated and deactivated. This function simplifies the processing of the simulation models.

For example, if one signal is instantiated in two different charts and configured with different components in each case, the deactivated chart is not taken into consideration during the consistency check. The function of the corresponding component in the deactivated chart is therefore not implemented and simulated.

Charts can only be activated or deactivated if the simulation is not running and the corresponding project is not write-protected.

Activation or deactivation of the charts is possible if an individual chart or coupling is write-protected.

Deactivated charts are still taken into consideration by the "Find & replace (Page 323)" and "Analysis (Page 329)" functions.

Two charts with the same name are still only permitted to be created in different folders.

Deactivated charts continue to be taken into consideration with regard to License model (Page 967).

All charts in a folder can be deactivated by deactivating the folder.

A deactivated chart is marked with the following symbol.



3.2.5.2 Activating charts

Requirement

- The project is not write-protected.
- A chart is deactivated.
- The simulation is not running.

Procedure

1. Select the chart you want to activate in the project navigation or project manager.
2. Select "Activate" in the shortcut menu.

Result

The selected chart has been activated.

See also

Find & replace (Page 323)

Basics of the analysis function (Page 329)

3.2.5.3 Deactivating charts

Requirement

- The project is not write-protected.
- The simulation is not running.

Procedure

1. Select a chart you want to deactivate in the project navigation or project manager.
2. Select "Deactivate" in the shortcut menu.

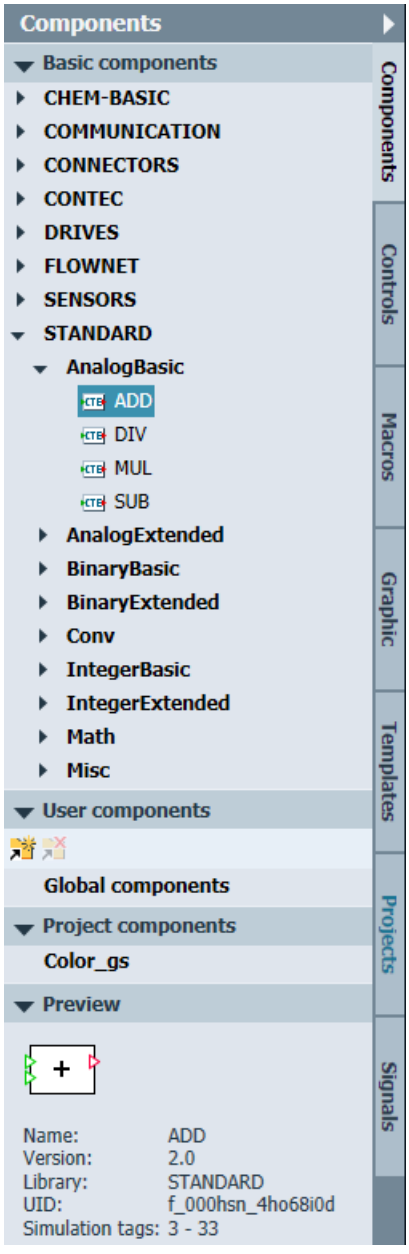
Result

The selected chart has been deactivated.

3.3 Task cards

3.3.1 "Components" task card

When SIMIT is started, the basic library, the global library and the project library are read in and displayed in the "Components" task card. Any library directories in the "User Components" area that were open when SIMIT was last closed are opened again.



The task card comprises the following areas:

Basic components

This is where you will find the component types of the basic library. The basic library is created when SIMIT is installed. The component types in the basic library cannot be changed and no component types can be added. The component types in the basic library can, however, be saved as copies in the "User components" and "Project components" areas.

You can find additional information on this in section: Basic library (Page 419).

Note

Adding components used in the project to a library

You can insert components used in the project in the following areas with drag & drop:

- User components
- Project components

User components

This is where you will find the component types that you have created or that other people have made available to you. If you store component types in the "Global components" area, they will be available in every SIMIT project.

You open any library directory with the command and have access to the component types stored in it. The command is used to remove a selected directory from this section.

If you create user component types with the CTE, you must save them either in a library directory or in *Global components* so that they are available in SIMIT. SIMIT updates the "User components" area automatically if you save a component type there with the component type editor.

You can find additional information on creating user components in the "SIMIT – Component Type Editor" help.

Project components

This is where you find the component types that you can use with the open SIMIT project. If you archive the project, all component types in this section are archived with the project as the project library. If the project is then retrieved, these component types are also available again.

You can move component types anywhere within the two "User components" and "Project components" sections or add them as copies. The component types of the "Basic components" section can only be added to the other two areas as copies.

Open component types with the "Open" command in the shortcut menu.

Preview

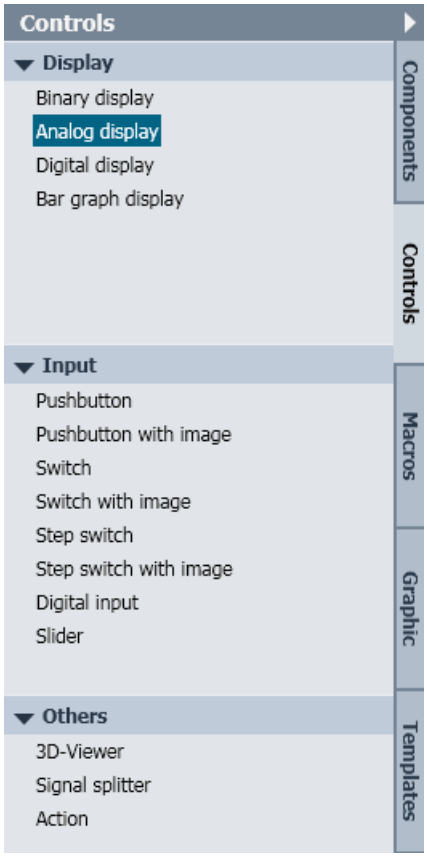
The following information is displayed for a selected component type in the preview for the "Components" task card:

- **Symbol**
  The basic symbol of the component type.
- **Name**
  The name entered in the component type.

- **Version**
  The version information entered in the component type.
- **Library**
  The library information entered in the component type.
- **UID**
  The unique identification automatically assigned to the component type.
- **Simulation tags**
  The information entered in the component type regarding the number of simulation tags.

3.3.2 "Controls" task card

Controls display and represent signal values.



The task card comprises the following areas:

Display

You will find the following controls for displaying signal values here:

- Binary display (Page 584)
- Analog display (Page 585)

- Digital display (Page 586)
- Bar graph display (Page 588)

Input

You will find the following controls for entering signal values here:

- Pushbutton (Page 590)
- Pushbutton with image (Page 591)
- Switch (Page 592)
- Switch with image (Page 593)
- Step switch (Page 594)
- Step switch with image (Page 595)
- Digital input (Page 596)
- Slider (Page 598)

With the "with image" option, you can add images to the control on the chart. For example, photographs give you a realistic representation of the controls.

Others

Here you will find the following additional controls:

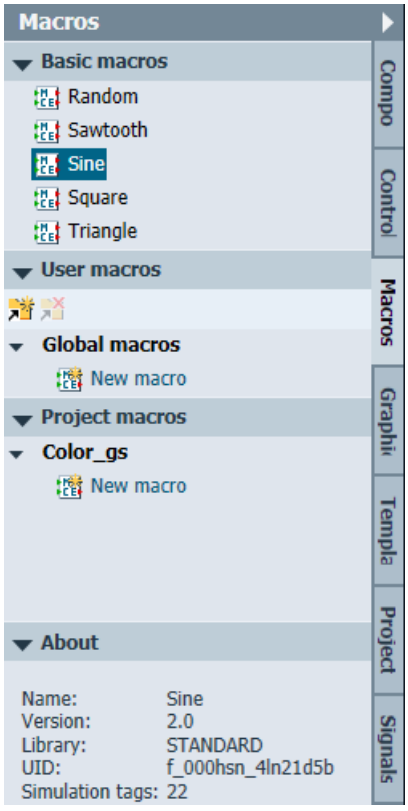
- 3D Viewer control (Page 602)
- Signal splitter (Page 600)
- Action (Page 602)

The controls can be linked to all signals in the "Signals" task card. You can use controls to display not just the input and output signals of couplings and components, but also status values of components, and to set editable parameters while simulation is in progress.

You can find additional information in section: Visualizing signals (Page 257).

3.3.3 "Macros" task card

You can combine repeated functions of a simulation model in a macro component. Using the macro then provides simple access to this function: You now longer need to copy this function from a chart and paste it to other charts: you can simply drag the macro component from the task card in the same way as the component types, configure it and link it to other components or macro components.



The task card comprises the following areas:

Basic macros

Here you find five macros that can be used as signal generators. They can be configured with respect to time slice duration and amplitude and yield different signal patterns:

- Random
- Sawtooth
- Sine
- Square
- Triangle

Note

Adding macros used in the project to a library

You can insert macros used in the project in the following areas with drag & drop:

- User macros
- Project macros

User macros

You can create your own macro components here. These macro components are saved in the work area of SIMIT and are therefore available for all projects. You use the "" symbol to open the folders that contain the macro components. You use the "" symbol to close a selected folder and remove it from this area.

Project macros

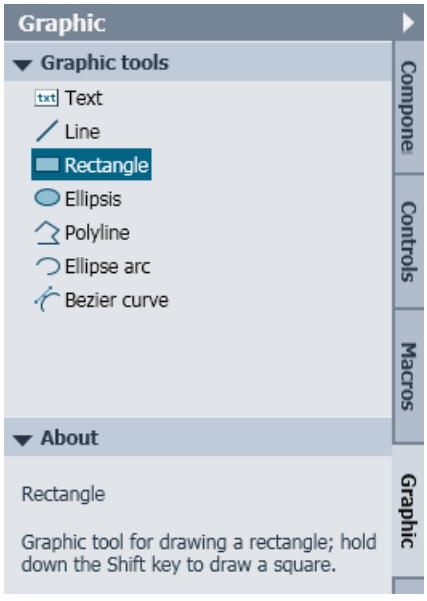
You can create your own macro components here. These macro components are stored in the project folder. They are only available while this project is open. All macro components stored here are archived with the project. Therefore they are available in the project again when you retrieve the project.

Info

Here the name, version, library, UID and number of simulation tags of a selected macro component are displayed.

3.3.4 "Graphic" task card

The task card "Graphic" contains graphic elements for use in charts.



Graphic elements

The various graphic elements are listed.

Drag-and-drop the required graphic element to the chart editor, where it is displayed and can be edited. Symbols from the chart editor are available for editing.

You can find additional information in section: Visualizing a simulation (Page 26).

Info

You will find a brief description of the selected graphic element here.

3.3.5 "Templates" task card

The "Templates" task card is available when the Project manager is open.



The task card comprises the following areas:

Basic templates

The SIMIT templates are located here.
You can drag these basic templates to the "User templates" or "Project templates" areas and edit them there.

User templates

Save the templates for your SIMIT installation here. These templates are then displayed in the "Global templates" section. User libraries can be opened with the "" symbol and closed with the "" symbol.

Project templates

The templates stored here are saved in the project and are archived with the project. After opening or retrieving a project these templates are available again.

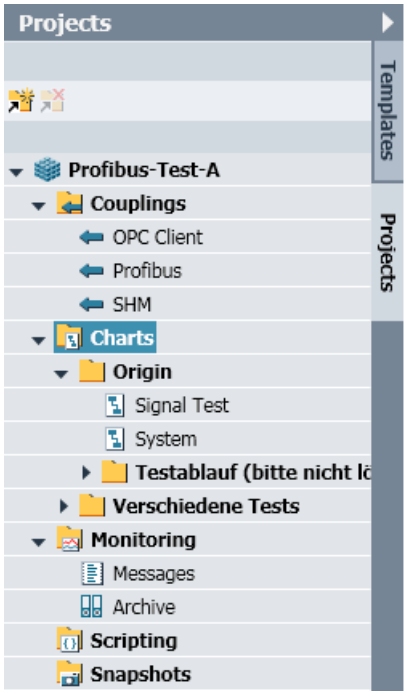
You can drag and drop or copy defaults within "My templates" and "Project templates".

You can find more information in Templates (Page 280).

3.3.6 "Projects" task card

More than one project is open at once in the "Projects" task card. This allows you to put together a project using several "Standard projects", for example.

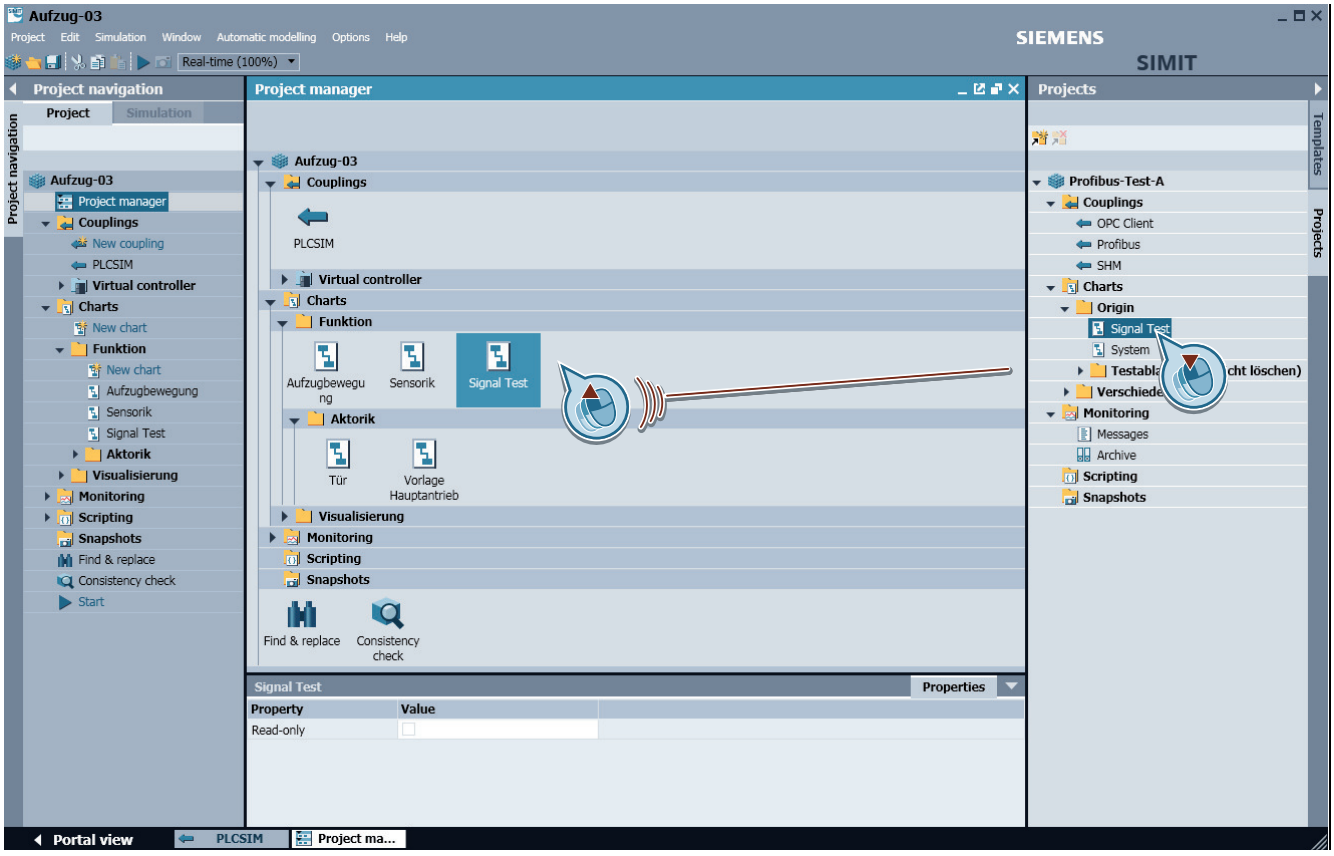
The "Projects" task card is available when the project manager is open. The project manager is also available when simulation is running. Project elements can then also be opened from the project manager.



Example:

1. Retrieve each of the 2 projects to project folders.
2. Open another project in SIMIT.
3. Open the project manager.
4. Click on the "" symbol in the "Projects" task card.
   The "Open project" dialog box opens.
5. Navigate to the project folder of the open project and select the project file *.simit.
   The project opens in read-only mode in the "Projects" task card.

You can now drag-and-drop individual elements such as charts or complete folders to the project manager to create a copy in the current project as shown in the figure below:

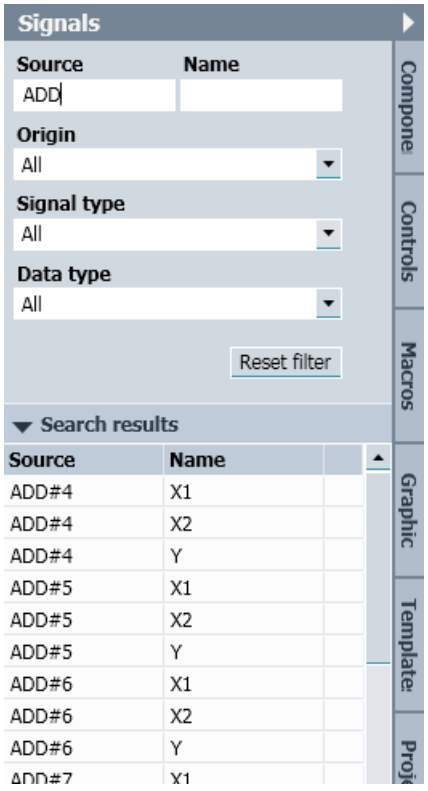


You may also open individual charts from a project in the chart editor, copy parts of these charts and paste them into charts in another project.

Close the projects in the task card by clicking on the "" symbol.

3.3.7 "Signals" task card

In the "Signals" task card, you can search for the signals in the open project and have them displayed.



You can select the following search criteria:

Source
Enter any text as search criterion here.

Name
Enter any text as search criterion here.

Origin
Select a search criterion from the drop-down list.

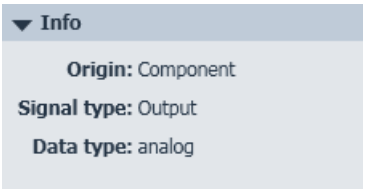
Origin
Select a search criterion from the drop-down list.

Data type
Select a search criterion from the drop-down list.

You can use the "Reset Filter" button to clear the settings and display all the signals of the open project under "Search Results".

Not all search criteria need to be assigned. One search criterion is enough to display search results.

Info



If you have selected a signal, the origin, signal type and data type are displayed in the "Info" area.

You can find additional information in the following sections:

- I/O signals for connection on charts (Page 230)
- Addressing a signal symbolically (Page 237)

3.4 Macro component editor

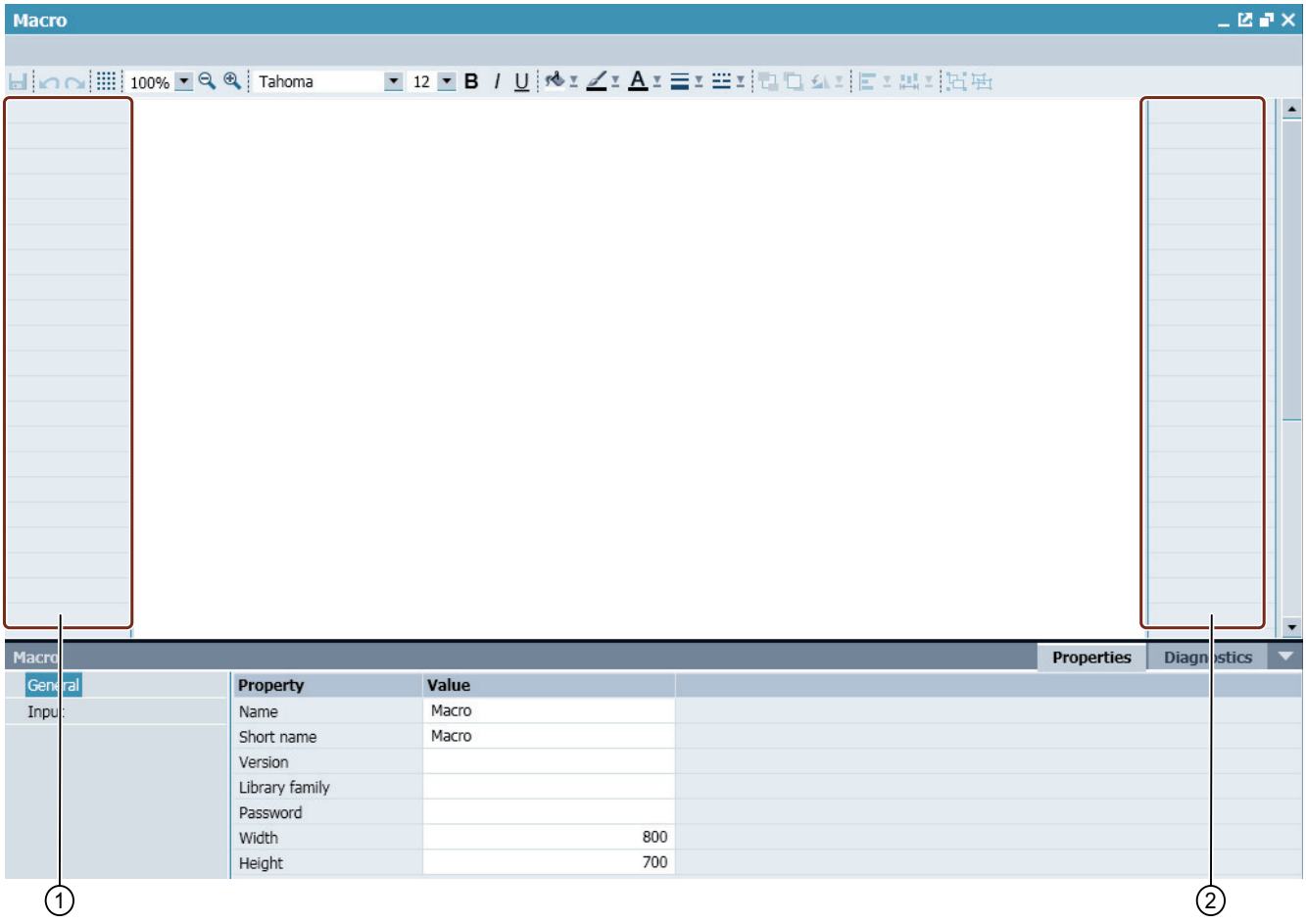
3.4.1 Macro editor

You can combine repeated functions of a simulation model in a macro component.
You can find additional information in section: "Macros" task card (Page 265).

Proceed as follows to create a new macro:

1. Open the "Macros" task card.
2. Double-click on the " New macro " symbol in the "User macros" area.
   A new macro component is created in this area.
3. Assign the new macro component a name or accept the default.
   The macro editor opens automatically when you confirm the default or changed name with the Enter key.

The only difference between a macro chart and other charts is the two sheet bars on the left and right:



① Sheet bar for inputs
② Sheet bar for outputs

The inputs and outputs of the macro are assigned to the cells in the sheet bars:

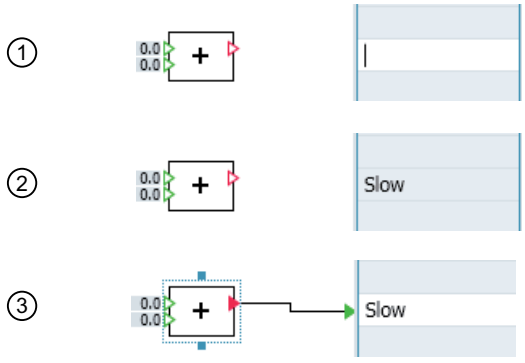
- Inputs in the left sheet bar
- Outputs in the right sheet bar

As with charts, you can use all components in the library and static graphics to model certain responses in the macro component. Of the connector components, you may only use the global and the topological connector.
You can find additional information in section: Topological connectors of macro components (Page 275).
Controls and animations are not available in macro components. Macro components cannot be hierarchically structured, which means you cannot insert any macro components into a macro chart.

If you have created a chart already and want to use it to create a macro component, you can copy the relevant parts of the chart and paste them to a macro chart. Only usable components are pasted to the macro chart.

Proceed as follows to add I/O to the macro component:

1. Double-click on a cell in the sheet bar.
   The cell opens for entering a name, as shown in the figure below under ①.
2. Enter a name.
3. Confirm your entry with the Return key.
4. Connect the macro I/O to the I/O of a component, for example by dragging the component I/O to the macro I/O, as shown in the figure below under ②.
   The connection is then shown by a connecting line (③).



5. Follow this procedure for all I/O.
6. Save and close the macro chart.

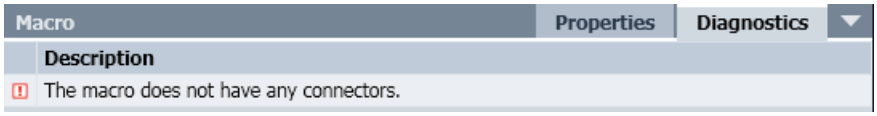
Alternatively, you can also generate a macro component I/O by clicking on the connection at the component I/O with the mouse to open it, dragging it to the required cell in the sheet bar and closing it with another click. The name of the component connector is used as the default in the sidebar.

Note

Every connector must have a unique name. The name must contain only letters, digits and the underscore character, and must start with a letter. The name is case sensitive.

Macro components containing errors are marked with the "" symbol.

Macro components with errors can be opened and edited in the editor, but you cannot use them in charts. You can find an explanation of the errors on the "Diagnostics" tab of the editor property view.

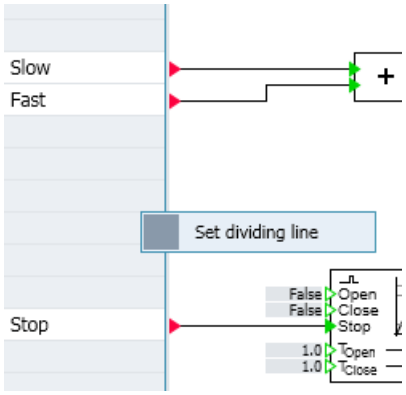


3.4.2 Inserting dividing lines between macro component I/Os

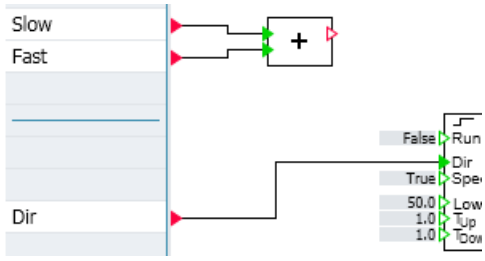
The I/Os of macro components are set one after the other in sheet bars. Insert dividing lines to separate the I/Os visually.

Follow these steps:

1. Open a macro component in the macro editor.
2. In one of the two sheet bars, select the shortcut menu command "Set dividing line" between two I/Os:

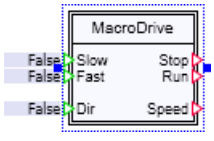


The dividing line is now shown in the cell as a line:



Display in the symbol of the macro component

The I/Os are now also offset from one another in the macro component symbol. This is represented by a blue rectangle. In the figure below, you can see that the outputs have also been offset from one another using a dividing line:

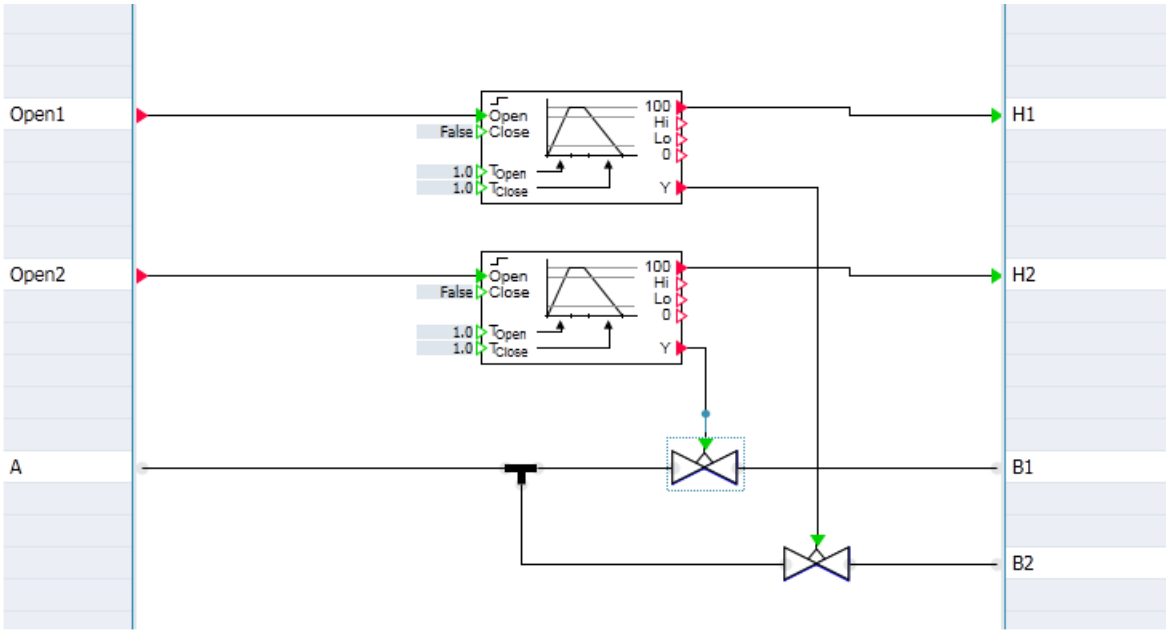


Deleting dividing lines

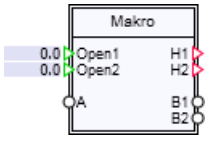
Dividing lines can be deleted using the corresponding command in the shortcut menu of the cell.

3.4.3 Topological connectors of macro components

Components from the FLOWNET and CONTEC libraries can be used in macro components. Open topological I/Os from these components are mapped to topological I/Os in the sheet bars of the macro component.



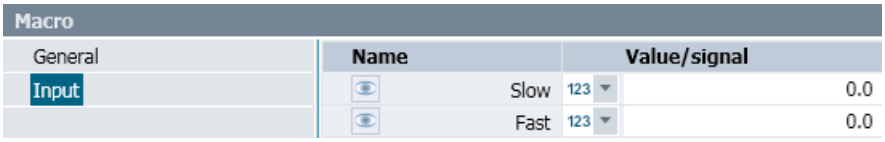
If macro components of this type are used in charts, the topological I/Os of the macro component can be connected to topological I/Os of other components or other macro components. The topological structures defined in the macro components are in this case merely substructures of the overall topological structure of a flow or transport network.



3.4.4 Default settings for macro component inputs

Default settings can be applied to individual inputs of macro components. Follow these steps:

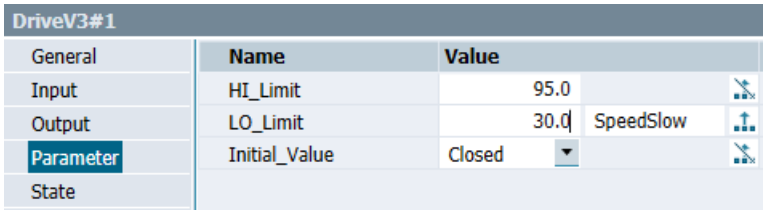
1. Select the "Input" entry in the property view of the macro component editor.
2. Enter the required values in the "Value/signal" column.



3.4.5 Defining parameters of macro components

You can specify parameters for macro components. The only parameters available are, however, those of the components used within the macro component.

Component parameters are displayed with the "" symbol in the property view of the macro component.



Editing parameters

- Click on the "" symbol to set a parameter as a parameter of the macro component. This moves the parameter from the component level to the macro component level.
  The symbol changes to "".
- In the text box to the left of the "" symbol, enter a name for the parameter (example in the screenshot above: "SpeedSlow").
  The parameter is then displayed in the macro component with this name.
- In the text box below the "Value" column, enter a suitable default setting (example in the screenshot above: "30.0").

When the macro component is used in a chart, the parameters and their default settings are now shown in the property view.

3.4.6 Properties of macro components

The following properties are available in the property view of the macro component editor:

- **Name**
  The name of a macro component corresponds to the file name under which a macro component is saved in the file system (work area or any directory in the file system). The macro component is displayed with this name in the "Macros" task card.

  Note

  If macro components have been imported from an earlier SIMIT version, the file name may initially differ from the macro name.

- **Short name**
  The short name is displayed in the header in the instance of a macro component.
- **Version**
  The version of a macro component can be freely assigned. It is displayed in the "Info" section on the "Macros" task card and in the tooltip for the macro component in the chart.
- **Library family**
  The library family of a macro component can be freely assigned. It is displayed in the "Info" section on the "Macros" task card and in the tooltip for the macro component in the chart.

- **Password**
  Macro components can be password protected. If a macro component is password protected, it can be used on charts but cannot be opened without entering the correct password. The last password entered to open password protected macros is saved until the project is closed.

  Note

  Keep the passwords for your macro components in a safe place, otherwise you will not be able to open even your own password-protected macro components.

- **Width**
  The width of the symbol.
- **Height**
  The height of the symbol.

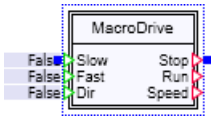
3.4.7 Find and Replace in macro components

Select the "Find & replace" command in the shortcut menu of the macro component to replace components within a macro component.

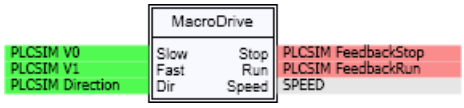
The commands opens the "Find & replace" editor with a preset focus on the macro component.

3.4.8 Using macro components

Like other components, macro components can be inserted in a chart with drag-and-drop. It is displayed on the chart as a rectangle with a double frame. The name of the macro component is displayed in the symbol header. The inputs and outputs are arranged on the left and right hand sides of the component in the sequence specified in the macro chart.



You can use a macro component on a chart like any other component. It can be connected with components and controls as well as with other macro components.



By double-clicking the macro component in the diagram, you open the macro chart in read-only mode, which means you cannot make any changes to the macro chart.

3.5 Migrating projects from previous versions of SIMIT

Projects from SIMIT V7.x can be opened or retrieved unchanged in the current version of SIMIT.

Note

Retrieving or opening a project of an old version of SIMIT may take a very long time the first time it is done.

Projects from SIMIT versions older than SIMIT V7.x cannot be used. Such projects must first be opened with a SIMIT V8 version and migrated.

Automatic model creation

4

SIMIT provides a range of automatic mechanisms for generating and editing charts. The automation is based on files that have a format that can be read by SIMIT. The files contain the information needed to generate or modify charts. This information is automatically converted into charts when the corresponding file is imported.

The following automatisms are available:

- **Templates**
  Templates are patterns for charts in which placeholders are defined for parameters, defaults, etc. When a table is imported in which values are set for the placeholders, the charts are automatically generated from the templates. IEA files exported from PCS 7, CMT files and Excel tables are based on templates.
  You can find additional information on this in section: Templates (Page 280).
- **Table import**
  Import of files in table format. The tables contain the replacements for the placeholders.
  You can find additional information on this in section: Table import (Page 287).
- **IEA import**
  Import of an IEA file from PCS 7. A corresponding chart is generated for each CFC based on a model or process tag type.
  You can find additional information on this in section: IEA import (Page 291).
- **CMT import**
  Importing control module types from PCS 7.
  You can find additional information on this in section: CMT import (Page 293).
- **Automated import**
  Import of charts from a ZIP archive or an XML file.
  You can find additional information on this in section: Automated import (Page 299).
- **Create device level**
  Instead of an external file, this automatism uses information available in the parameters of the components used in the simulation model. This allows you to instantiate templates automatically for certain types of component, for example to create the right control logic simulation for process technology devices.
  You can find additional information on this in section: Generating the simulation of drives and sensors (Page 881).
- **Bulk processing of parameters**
  Export and import of parameters and input defaults from and to existing charts.
  You can find additional information on this in section: Exporting data (Page 305).

The automatic mechanisms are listed under the "Automatic model creation" menu item. If they are accessed from there, the new charts are created at the top project level or, if you specify a folder hierarchy, relative to the "Charts" system folder.

If the automatic mechanisms are called from the shortcut menu of a chart subfolder, all new charts are created relative to this selected folder. Since the parameter import does not create new charts, it is not included in the shortcut menu.

See also

Structure of the control file "procedure.cfg" for import from ZIP file (Page 304)

4.1 Templates

Templates prepare frequently recurring functions of a simulation model, for example parts of a chart, in such as way that they can be used as a template for charts when you create new projects. You can use all of the same elements in templates that you use in charts:

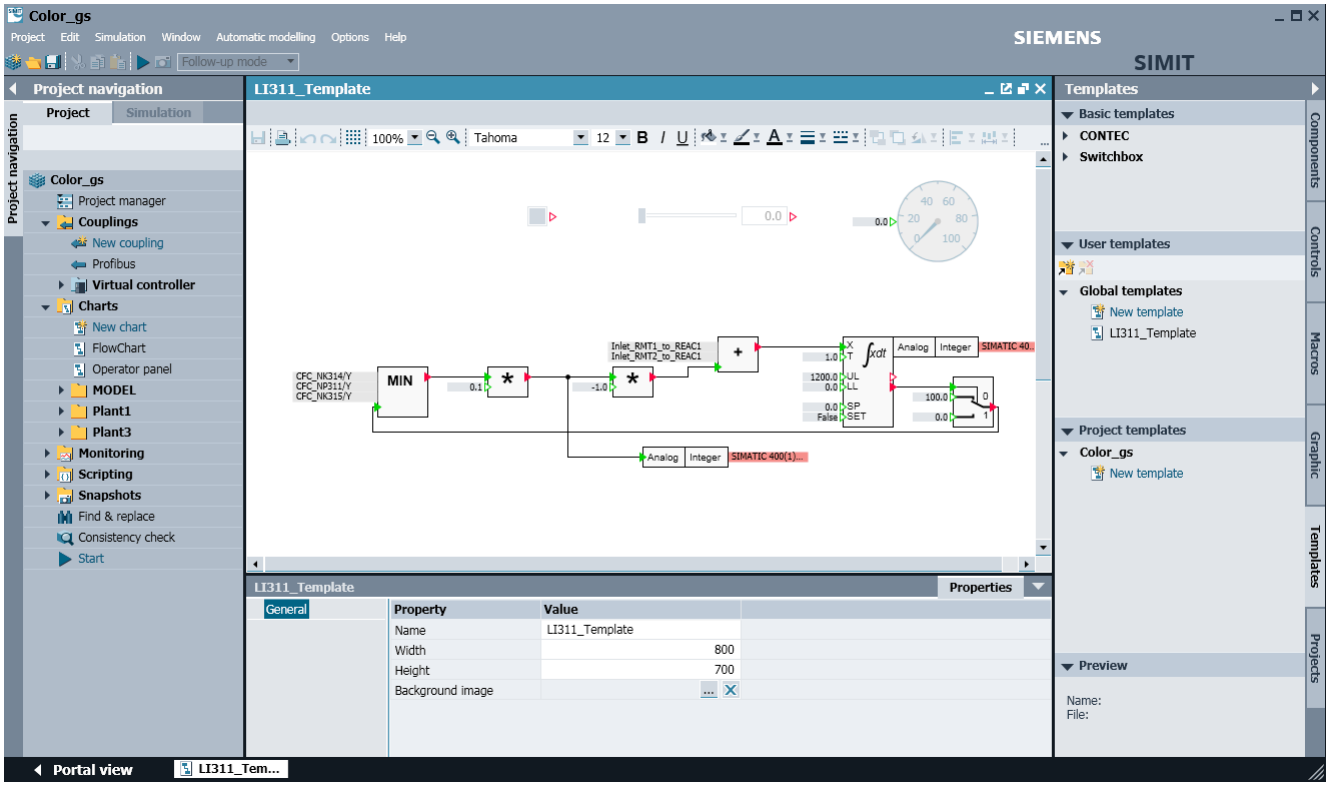
- Components
- Macro components
- Controls
- Graphics

In templates, unlike in charts, placeholders are used for various elements in the components in the template. When a template is instantiated, a chart is generated in which the placeholders are replaced with values, signal names, etc. You have the following options to instantiate templates:

- Drag-and-drop the template to the project
- Importing files
- Using the simulation model

Proceed as follows to create templates:

1. Open the "Templates" task card in the project manager.
2. Open the template dialog to create a new template by double-clicking on " New template ".
3. Open a template for editing by double-clicking on the template in the task card.
4. Edit the template with the components, controls, macro components, graphics, templates and projects available to you in the template editor resources.



Templates in the "Basic templates" area can only be opened for viewing in the template editor; they cannot be edited. To indicate this the template is opened in the editor with a white title bar. If you wish to edit a basic template, copy it to the "User templates" area.

The difference between templates and charts is that placeholders are used in templates.

The placeholder symbol "" allows you to define as placeholders: all inputs and parameters of components and macro components, signals in controls and animations, and component names.

For some controls you can also define the type, default setting and scaling as placeholders. The table below shows which properties can be used as placeholders for controls.

Table 4-1 Placeholders for controls

Control	Name	Time slice	Type	Default setting	Scaling	Connec-tor	Objec-tive
Button	X	X	X				
Button with image	X	X	X				
Switch	X	X	X	X			
Switch with image	X	X	X	X			
Step switch	X	X		X			
Step switch with im-age	X	X		X			
Digital input	X	X		X			
Slider	X	X		X	X	X	
Bar graph display	X	X			X	X	

Control	Name	Time slice	Type	Default setting	Scaling	Connec-tor	Objec-tive
Binary display	X	X				X	
Analog display	X	X				X	
Digital display	X	X				X	
Action	X						X

In some cases the type and default setting of controls take the form of language-dependent lists in the user interface. If you define these properties as placeholders, you must specify the position of the corresponding term in the list when making the replacement.

Table 4-2 Parameter value for the NC/NO contact type

Type	Value
NC contact	0
NO contact	1

Table 4-3 Parameter value for the Off/On default setting

Default setting	Value
Off	0
On	1

4.1.1 Find and replace in templates

Select the "Find & replace" command in the shortcut menu of the template to replace components within a template.

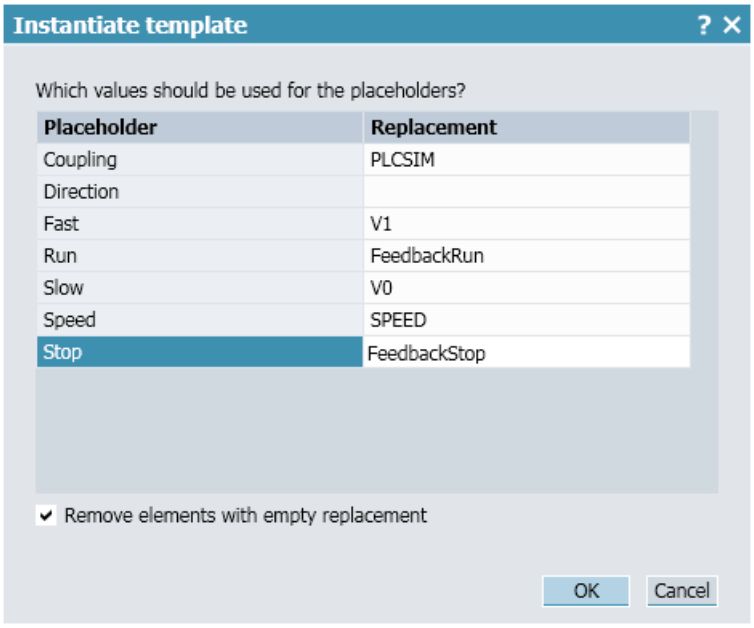
The commands opens the "Find & replace" editor with a preset focus on the template.

4.1.2 Instantiating templates by entering replacements

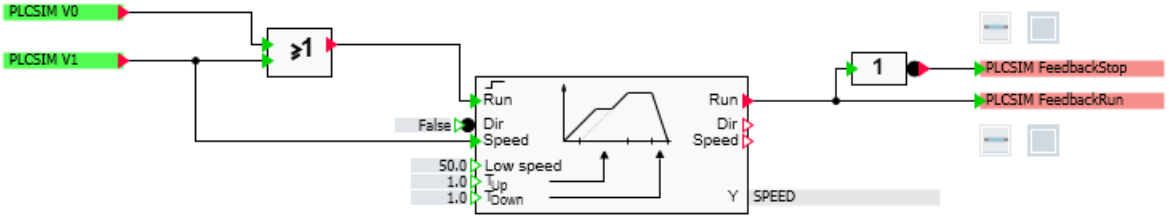
Templates are instantiated by replacing placeholders with appropriate values. Components that have a name not defined as a placeholder, receive the name they have obtained on the template, possibly supplemented by a sequential number, which makes the component name unique.

In the Project Manager, you may instantiate a template by dragging and dropping it from the task card into a project folder. Open the project manager and select the Templates task card. If you now drag the template that was created above, a dialog appears as shown in the figure below. A two-column list is displayed, which shows all of the placeholders that are contained in this template. Please enter the replacement for each placeholder in the right column.

If the "Remove elements with empty replacement" option is checked, all elements are removed where a placeholder does not have a replacement in the table. Global connectors and I/O connectors, for example, are deleted if no signal is specified. By way of example, this is the case for the Direction placeholder in the figure below. You may thus create your templates in such a way that they cover several use cases, and may then instantiate the template for a specific use case by not replacing the placeholders that apply to the other use cases.



After confirming the dialog you will see a new chart with the same name as the template in the Project Manager. When you open the chart, you will see the instantiated simulation. It is identical to the simulation of the main drive, except for the missing output signal *Direction*. The output connector was not instantiated because it lacked a replacement. If you had deselected "Remove elements with empty replacements" in the instantiation dialog, a connector with a signal value i.e. an empty connector would have been generated in the instance.



4.1.3 Creating tables from a template

If you want to create a table for a template so that this template can then be instantiated multiple times, SIMIT allows you to create a table which already contains a title row matching the template. Select the "Export to Excel" command from the shortcut menu of the template.

The next steps will depend on whether or not Microsoft Excel is installed on your computer.

- **Microsoft Excel installed**
  A new workbook is directly opened in Excel, with the entries needed for the selected template already entered in the top row on the first sheet. Complete the table and save it in a location of your choice for use in table import.
- **Microsoft Excel not installed**
  An Excel file in *.xlsx format is created, with the entries needed for the selected template already entered in the top row on the first sheet. Save this file in a location of your choice for editing in Excel later – if necessary on another computer.

The first three columns of the generated tables contain the terms HIERARCHY, TEMPLATE and CHART. The subsequent columns contain the names of the placeholders used in the template. The name of the template is already entered in the second row, but you must enter all other information yourself.

	A	B	C	D	E
1	HIERARCHY	TEMPLATE	CHART	GATEWAY	NAME
2		Rail-S4			
3					
4					

Note

Please note that when this table is imported, the profile is set to "Placeholders defined in 1st row".

4.1.4 Defining a folder hierarchy for templates

You can specify the folder in which to search for a template in your tables. For the TEMPLATE placeholder, enter the complete folder hierarchy as shown in the "Templates" task card. Use the backslash '\' as the separator. To refer to a template that is not located in a subfolder, enter the name of the template preceded by a backslash.

If the template name is entered without separators, all folders and subfolders are searched for a template of this name as before. In this case, make sure that there are not two templates with the same name in different subfolders. The three panes of the "Templates" task card are searched in the following order:

1. Project templates
2. User templates
3. Basic templates

4.1.5 Addressing a module via the I/O address

The identity of a module on PROFIBUS or PROFINET is determined by the slave or device number and the slot number. Accordingly, both values must be entered in the unit connector if it is used to identify a device.

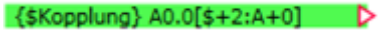
If such devices are used on templates, this information is not always available. Instead the module start address may be known. Therefore it is now permissible to identify a module on templates by its start address, for example, "EW512" or "AW512" instead of "[1][4][4]".

When the simulation is started it is necessary to know the address with the slave or device number and the slot number. Therefore when a template is instantiated, a start address is automatically converted to the address with the slave or device number and slot number. If there is no coupling with the appropriate address when the template is instantiated, you can also run the automatic conversion later with the command "Options > Assign coupling signals (Page 981)".

4.1.6 Indirect addressing

In the input and output connectors, an additional item of information can be included on templates, which allows access to signals located relative to a known address. This expression has the form [$+n:RW], where:

- `n` : is the offset to be added in bytes
- `R` : is the input or output identifier E, I, A or Q
- `W` : is the data width, B, W or D
  If an individual bit is being addressed, the bit address is specified in the form "+m". The bit position can be shifted by 0 to 7 places with "m".
  For example, if a quality bit for a known binary signal is being calculated, which is located two bytes behind the user data, the following expression might be used:

  That is resolved to:

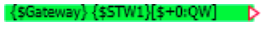
  

So, for example, if the address of a control word EW560 is known, the associated status word AW560 can be specified as follows:

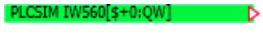
IW560[$+0:QW]

Or, if the control word EW560 has the symbolic name STW1 in the coupling, it can also be specified in the form STW1[$+0:QW]

If the coupling already exists when the templates are instantiated, an attempt is made to replace the expression with the corresponding signal. If not, this expression is retained even after the signal to which it relates is replaced. For example, if the control word STW1 exists in a PLCSIM coupling with its address IW560 but no associated status word, then the connector



is replaced as follows:



However, you can also start complete replacement subsequently at any time by calling the function "Options > Assign coupling signals (Page 981)".

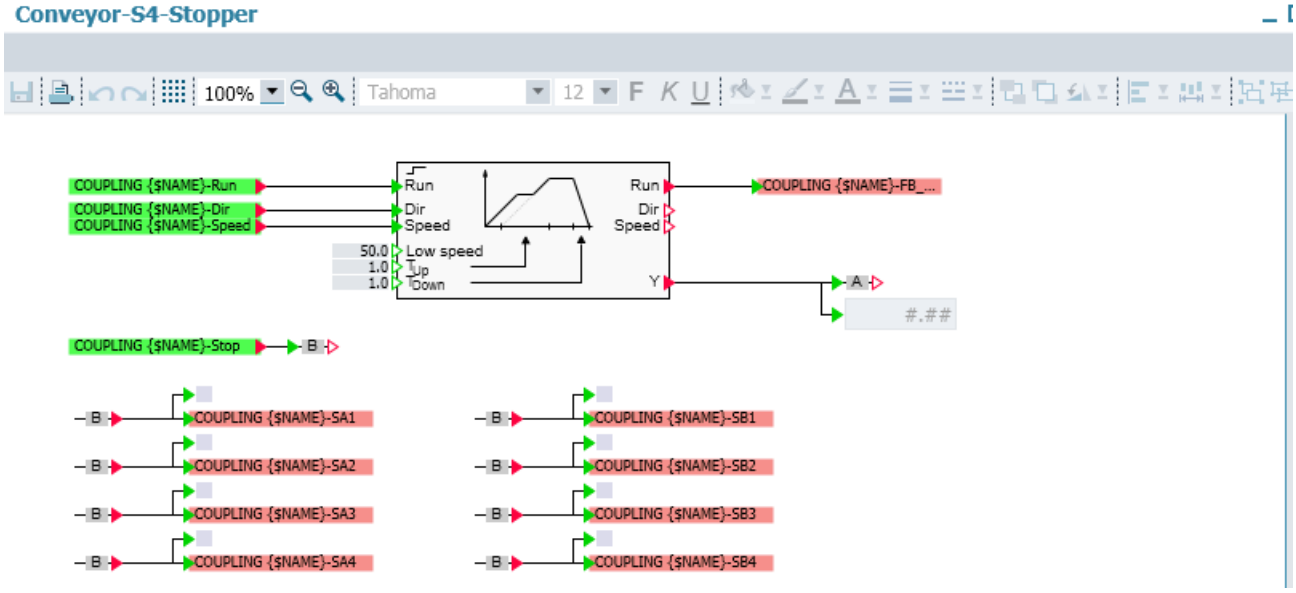
Note

Indirect addressing is only supported by the following couplings:

Virtual Controller, SIMIT SU, PRODAVE, PLCSIM and PLCSIM Advanced

4.1.7 Opening basic templates in the editor

Templates in the "Basic templates" pane can now be opened straight from the shortcut menu in the template editor. However, basic templates opened in this way can only be viewed, not edited. To indicate this the template is opened in the editor with a white title bar.



To edit a basic template, copy it to the "User templates" pane.

4.2 Instantiation of templates from files or the simulation model

4.2.1 Instantiating templates from files

Requirement

- SIMIT project is open.

Procedure

To instantiate the templates from files, follow these steps:

1. Select "Automatic model creation > Instantiate templates".
   The "Instantiate templates" dialog box opens.
2. Select the "Import file" option.
3. Enter the storage location of the import file.
4. To import tables, follow these steps:
   - To instantiate a template according to the table multiple times in a chart, select the grouping to be made in the chart: horizontal or vertical.
   - To line up the instances *horizontally*, enter the *maximum width* of the chart.
   - To line up the instances *vertically*, enter the *maximum height* of the chart.
5. To import IEA files, follow these steps:
   - Provide a name for the coupling to be used for I/O signals.
   - Enter a template that matches the import file.
   - Select the separator.
6. To import CMT files, follow these steps:
   - Provide a name for the coupling to be used for I/O signals.
   - Specify the folder in which the search for the SIMIT templates referenced in the CMT file is to take place.
7. If necessary, deactivate individual chart folders, charts or replacements under "Preview>>".
8. To start the import, click the "Import" button.

Result

The templates were imported and instantiated from a file. The generated charts are stored in the "Charts" folder.

See also

Table import (Page 287)

IEA import (Page 291)

CMT import (Page 293)

"Instantiate templates" dialog box (Page 995)

4.2.2 Table import

Templates can be instantiated using tables. The placeholders are always in the first row in these tables. These other rows contain the replacements for the placeholders and additional information about the templates to be used and the instances to be created.

In all tables, each row defines the instantiation of one template.

The following table formats are supported:

- ***.xls**
  Microsoft Excel format Office 97 – 2003. Only the first sheet of the workbook is taken into account. This file format is only available if Microsoft Excel is installed on your computer.
- ***.xlsx**
  Microsoft Excel format. Only the first sheet of the workbook is taken into account. This file format is available even if Microsoft Excel is not installed on your computer.
- ***.txt**
  Tab-separated list in text format. Through multi-selection, you can also group multiple text files and import them in one operation.

Note

If you still want to use files in *.csv format from projects in earlier versions of SIMIT, you must convert them to this tab-separated *.txt format or to the *.xls or *.xlsx format.

Note

When importing a file in *.xls or *.xlsx format that is still open in Excel, it is the last saved version of this file that is imported and not the currently open version.

Importing to SIMIT

The following options are available to perform a table import into SIMIT:

- In the portal view, select "Automatic model creation" > "Instantiate templates".
- From the Project view, select the menu command "Automatic model creation > Instantiate templates".
- Select "Automatic model creation" > "Instantiate templates" from the shortcut menu of a chart folder.

Placeholders in the first row

The meaning of the column is specified with column headers.

	A	B	C	D	E	F
1	HIERARCHY	TEMPLATE	CHART	P1	P2	P3
2	H1	T1	C1	1.0	E0.0	
3	H2	T2	C2	3.0		A1.0

The first three columns have a pre-defined meaning and their headings are also pre-defined:

- Column A (HIERARCHY) defines the folder hierarchy to be created in the project.
- Column B (TEMPLATE) defines the template to be used.
- Column C (CHART) defines the name of the chart to be instantiated.

Starting with column D, the table may contain any number of additional columns with replacements for the placeholders. The placeholder is given in the header of each column. In this table profile, there are no restrictions in the number of placeholders.

Note

You specify templates (*TEMPLATE*) with the full folder hierarchy. Use the backslash '\' as the separator. To refer to a template that is not located in any subfolder, enter the name of the template preceded by a backslash.

If the template name is entered without separators, all folders and subfolders are searched for a template with this name. Therefore, it is not permitted for two templates with the same name to exist in different subfolders. The three panes of the "Templates" task card are searched in the following order:

1. Project templates
2. User templates
3. Basic templates

Example of instantiation using a table

An example of a table can be found via Windows Start menu under "SIMIT" > "Samples" > "SMD". To instantiate the templates according to the table, select the menu command "Automatic model creation > Instantiate templates". The "Instantiate templates" dialog box opens.

Select the file "Sample.txt" as the table. To instantiate a template according to the table multiple times in a chart, you must specify how they are to be grouped and also how much space is available on the chart. To line up the instances *horizontally*, enter the *maximum width* of the chart. To line up the instances *vertically*, enter the *maximum height* of the chart.

If you now use the "Preview>>" button to open the preview, you will see the folder hierarchy and charts that are to be instantiated on the left-hand side of the preview. For each chart, you can see and verify the list of replacements according to the table in the right half of the preview before the charts are actually instantiated. To open the list, select the corresponding chart. You can now decide whether to accept all information from the table or whether to deselect certain replacements.



If a chart is deselected in the left tree view, it is not instantiated. By selecting a folder, you can also select or deselect multiple charts within it. You can tell from the check box whether a chart or folder is selected (✔) or deselected ( ) or whether at least one check box in one of the subfolders is deselected (✔).

In the right table, you can also select and deselect individual replacements. If you deselect a replacement, it is as if the replacement were not in the table at all.

4.2.3 IEA import

If you use PCS 7 and work with process tag types or models, create a suitable chart in SIMIT for any CFC that is based on a process tag type or a model. To do this, create the appropriate templates in SIMIT. Examples of templates can be found, for example, in SIOS.

If you have used one of the standard templates from a PCS 7 library in your project, you can use these sample templates and create a matching simulation. The procedure is explained below using the example of a motor drive (MOTOR). You can find two example IEA files from a PCS 7 project via the Windows Start menu under "SIMIT" > "Samples" > "SMD".

Importing to SIMIT

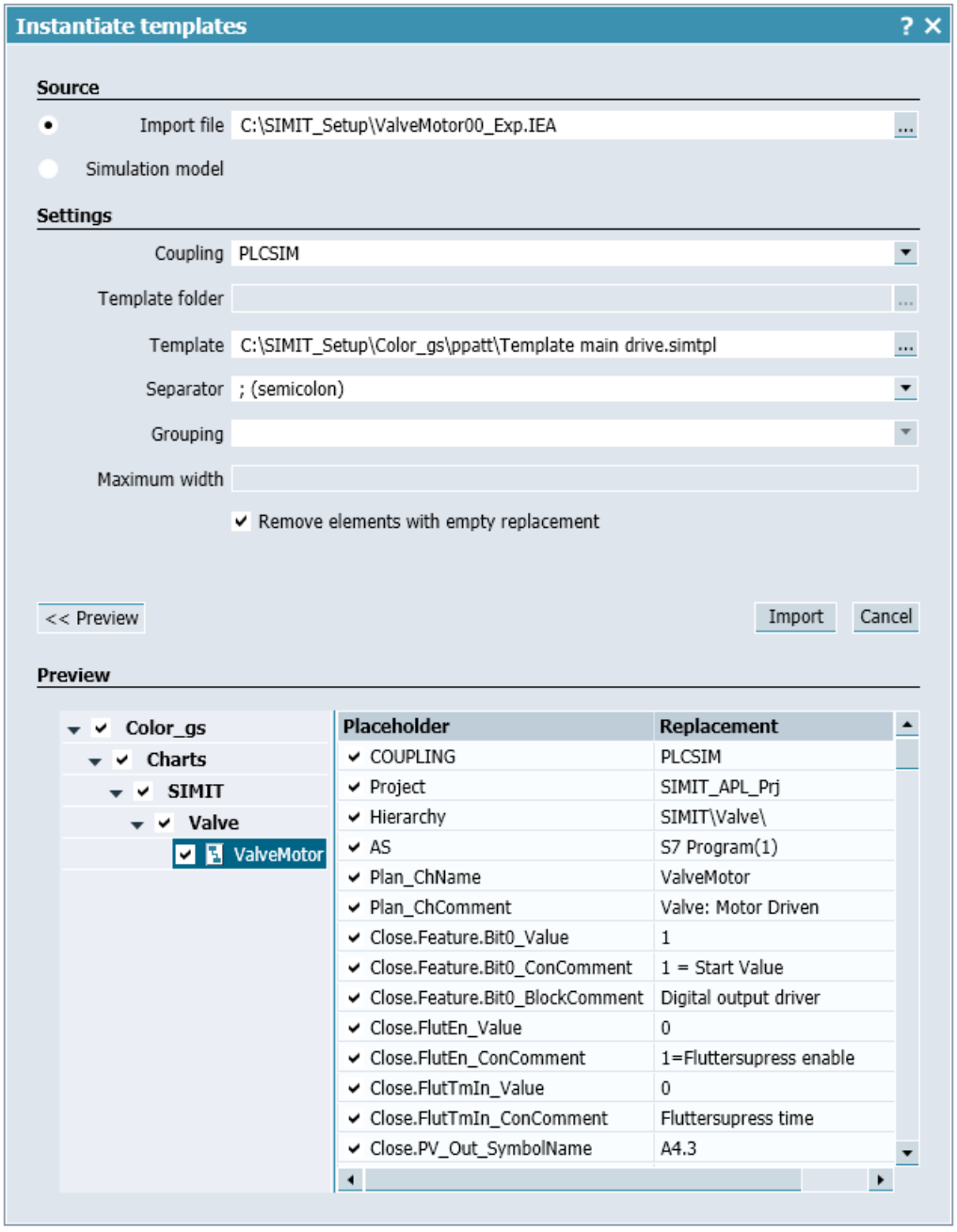
The following options are available to perform an IEA import into SIMIT:

- In the portal view, select "Automatic model creation" > "Instantiate templates".
- From the Project view, select the menu command "Automatic model creation > Instantiate templates".
- Select "Automatic model creation" > "Instantiate templates" from the shortcut menu of a chart folder.

The "Instantiate templates" dialog box opens.

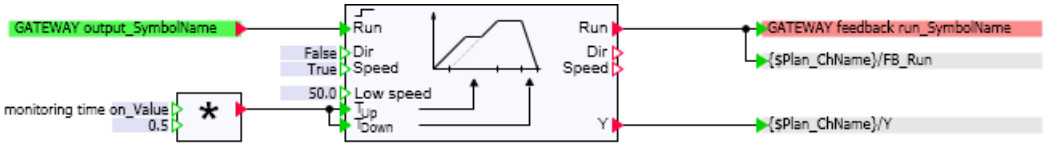
Provide a name for the coupling to be used for I/O signals.

In the preview, you will see that a chart is instantiated for each CFC. This chart has the same name as the CFC and is placed in a folder hierarchy that is identical to the folder hierarchy of the PCS 7 project. You can decide whether to copy all information from the IEA file or deselect specific replacements.

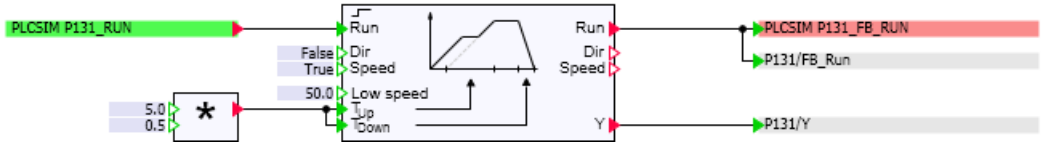


The following figure shows the template under (a) and the resulting instance under (b).

(a)



(b)



Now import the second sample file "VALVE00_Exp.IEA" for the VALVE template. This creates additional charts in the SIMIT project in the folder hierarchy that was created during the first import.

If you are using modified or user process tag types in your PCS 7 project, you can of course also use these IEA files with corresponding SIMIT templates. Just create matching templates, in "User templates", for example, or copy suitable basic templates and modify the copies.

See also

SIMATIC PCS 7 ES/OS System component updates (https://support.industry.siemens.com/cs/ww/en/view/63481413)

4.2.4 CMT import

CMT stands for "Control Module Type" in PCS 7. Control module types are used to create CFC templates and provide them with variable components. These CFC templates are then copied and individualized by instantiation. PCS 7 creates an exportable XML file for this, which includes all control module types as well as all instances in the project, their variable replacements and versions.

Export from PCS 7

This functionality is used in PCS 7 for bulk data engineering or for exchanging configuration information with other configuration tools, such as COMOS.

Note

The "XML Transfer" function required for PCS 7 is not included in the product package of SIMIT but can be obtained through SIOS (https://support.industry.siemens.com/cs/ww/en/view/63481413):

1. Choose category "Product Notes & Downloads".
2. Select PCS 7 version
3. "Engineering System Collection" download Vx.y.
   The "Engineering System Collection" contains several components, which you select via the installation wizard. You only need the "XML Transfer" component for exclusive use with SIMIT.

After the installation, the export can be performed directly from the PCS 7 project in the SIMATIC Manager (from the technological view) using the shortcut menu command "Export XML".

The exported XML file can now be imported into SIMIT to generate and test the instantiated control module types in the simulation.

Importing to SIMIT

The following options are available to perform a CMT import into SIMIT:

- In the portal view, select "Automatic model creation" > "Instantiate templates".
- From the Project view, select the menu command "Automatic model creation > Instantiate templates".
- Select "Automatic model creation" > "Instantiate templates" from the shortcut menu of a chart folder.

The "Instantiate templates" dialog box opens.

After the templates are instantiated, the charts can no longer be identified as "template instances", unlike in PCS 7. Therefore, it is not possible to subsequently switch versions or retain customizations. Synchronization is not possible.

If the project changes in PCS 7, the XML file must be exported again and imported once more into SIMIT. In this import, all the existing instances can be replaced with new charts. Individual customizations that were made after the last CMT import to the charts already instantiated are lost in the process.

Template creation

The following options are available in the Template Editor to define variables and placeholders in the templates:

- A variable can be assigned to the value of a property field. For this purpose, the "{$}" symbol to the right of it must be enabled. {$} means that the complete field content is interpreted as a variable name.
- If more than one variable or fixed and variable components are to be combined, you can also enter a variable name in the respective field using the notation {$ variable name}.

This enables you to specify replacements from the XML file in the instance of a template.

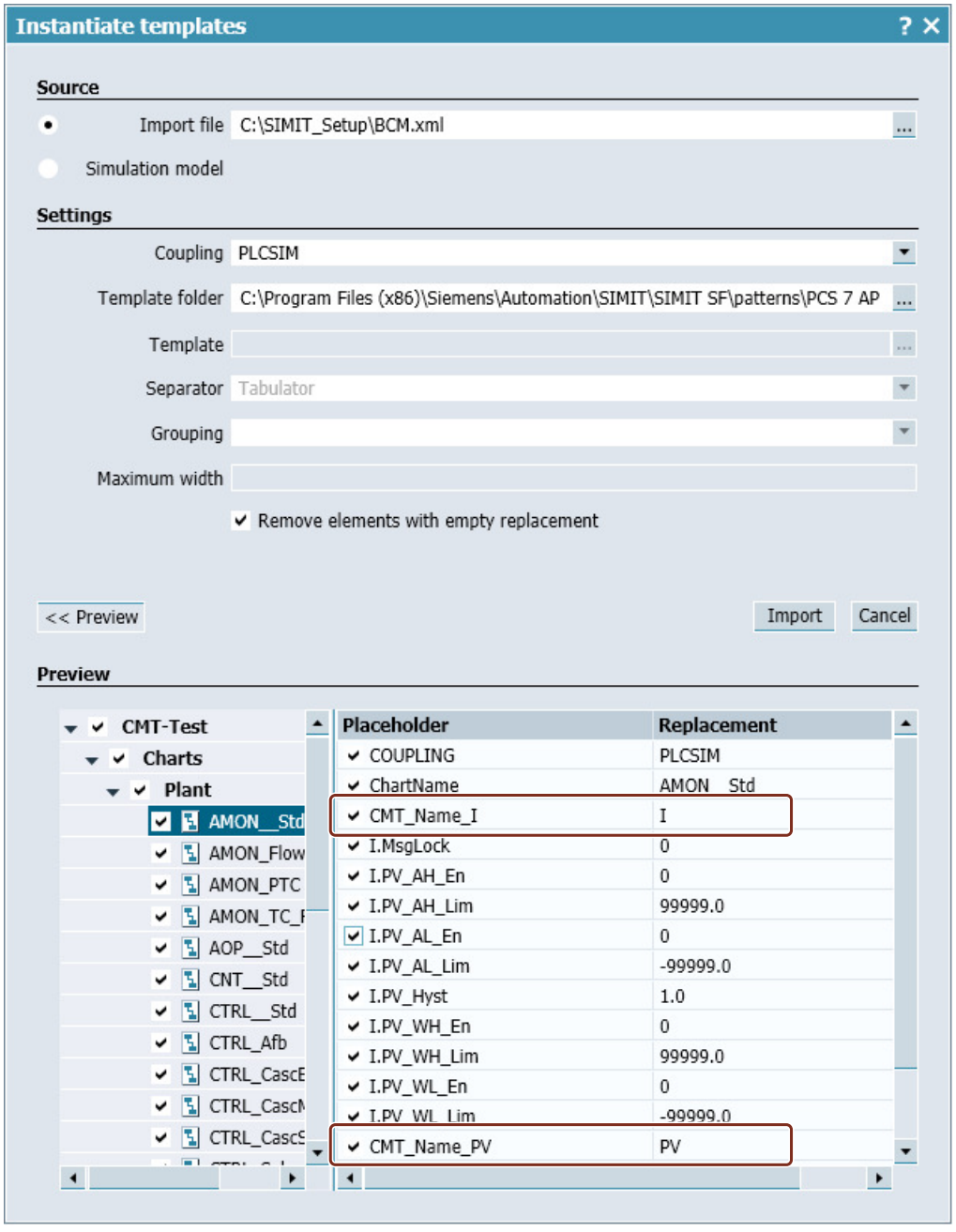
Versions and multiple connections in CMTs

The mapping of versions of a CMT is realized in SIMIT as follows:

- The "Remove elements with empty replacements" check box is selected.
- The variable "CMT_Name_x" is only in the list of replacements available when this variant is present in the CMT.
  This applies to all variables and parameters derived from this CMT.

Note

The "x" in "CMT_Name_x" can be seen as the CMT name and is "I" or "PV" in the figure below.



If a component is to be integrated in the simulation model in SIMIT with a dependency to an optional CMT in PCS 7, the name of the component, for example, must refer to this variable.

If the CMT is not in the XML file, its variables, parameters and signals are also absent and all components that contain one of these replacements are automatically omitted in SIMIT as well. This makes it possible to reflect the version concept of the CMTs in SIMIT.

Multiple interconnections to CMTs in PCS 7 can be implemented in SIMIT with connectors. Because these versions can be used only as an alternative in PCS 7, the mechanism of the versions described above can be used here in SIMIT to interconnect multiple outputs from components in the template to one input.

These components are assigned to the versions and their outputs are linked to the input of another component with connectors. If one of the two CMTs is now omitted in the instance, the corresponding component is also omitted in SIMIT and the connector remains unconnected on a chart. Such connectors do not exist for SIMIT which means there is no double assignment that would otherwise lead to a consistency error.

4.2.4.1 Instantiating templates for interconnection between CMTs

An interconnection between 2 CMTs via connectors is supported in the templates.

When an interconnection between 2 CMTs in a project will be reused, you can create project templates that contain this interconnection.

The "Instantiate templates" function offers the ability to set a placeholder for the creation of the connection.

Procedure

1. Create two project templates.
2. Place a desired CMT in each of the project templates.
3. Insert a connector to the desired input or output in each of the project templates.
4. Assign the connectors the same name. The name is used as a placeholder.
   The name of the placeholder consists of the following:
   - The type name of the CMT containing the signal input.
   - The name of the interconnected component containing the signal input.
   - The input name.

   The placeholder name is written in one word as follows:
   CMTTypeName_ComponentName.InputName.
5. Import the project files.
   Instantiating templates from files (Page 286)
6. The project templates are instantiated. The replacement the placeholder consists of the following:
   - The hierarchy of the CMT containing the signal input.
   - The instance name of the CMT containing the signal input.
   - The name of the signal input.

Result

The interconnection between the 2 CMTs was generated automatically.

4.2.5 Instantiating templates from the simulation model

Introduction

Instantiation from the simulation model uses parameters of the components used in the simulation model. Templates can be instantiated automatically for certain types of components, for example, to create appropriate simulation of the control logic for process engineering devices.

Requirement

- SIMIT project is open.
- The components in the project must contain the "TEMPLATE" and "HIERARCHY" parameters.

Procedure

To instantiate the templates from the simulation model, follow these steps:

1. Select "Automatic model creation > Instantiate templates".
   The "Instantiate templates" dialog box opens.
2. Select the "Simulation model" option.
3. Provide a name for the coupling to be used for I/O signals.
4. Specify the template folder in which the search for the referenced SIMIT templates is to take place.
5. To start the import, click the "Import" button.

Result

The templates were instantiated.

See also

"Instantiate templates" dialog box (Page 995)

4.3 Automated import

4.3.1 Import charts from ZIP or XML files

Introduction

The "Automated import" function supports the import of charts from the following sources:

- ZIP
  Imports charts with referenced components, even if the components are not contained in the SIMIT task card.
  Applications such as COMOS can generate these import files. An import file is a ZIP archive.
- XML
  Imports charts from an XML file. During import, existing templates for charts can be instantiated and custom charts can be created without the need for templates.
  The XML file contains a description of charts based on an XML syntax.

  Note

  In order to validate the XML file for correct syntax, run it through a validation with appropriate tools before importing.

There are three ways to carry out an automated import to SIMIT:

- In the portal view, select "Automatic modelling > Automated import".
- In the Project view, select the menu command "Automatic modelling > Automated import".
- In the shortcut menu of a chart folder, select "Automatic modelling > Automated import".

Requirement

- Project is open.
- A file suitable for automated import is available.

Procedure

Follow these steps to import charts from ZIP or XML files:

1. Start automated import using one of the above-mentioned commands.
   The "Automated import" dialog opens.
2. To import charts from a ZIP file:
   - Select "*.ZIP" as the file format.
   - Select the required file.

3. To import charts from an XML file:
   - Select "*.XML" as the file format.
   - Select the required file.
     The "Generic Import" dialog opens.
   - Select the required data.
4. Click "Import".

Result

The data is imported.

See also

Syntax of the XML file (Page 300)

Structure of the control file "procedure.cfg" for import from ZIP file (Page 304)

4.3.2 Syntax of the XML file

The file for import must be provided as a text file in XML format.

You can find the document type definition of the file to be imported in "Generic.dtd". You can find the file "Generic.dtd" under "SIMIT" > "Samples" > "xml" in the Windows Start menu.

A valid XML file starts with the element GENERIC. The version number 8.0 should be entered as the attribute for this element.

Element	Attribute	Description
GENERIC		*Folder*
	VERSION	In the document form described here, "8.0" must be entered as the version.

The generic import interface can also be used to create an arbitrarily nested folder hierarchy. The specified folder hierarchy in the import file is relative to the folder whose shortcut menu you have selected to perform the import. This is the "Charts" system folder when the "Automatic model creation" menu is used. If the import file contains no FOLDER elements, the templates are created directly in the selected folder.

Element	Attribute	Description
FOLDER		*Generic import interface*
	NAME	The name of a folder to be created. A deeper folder hierarchy can be achieved by nesting several FOLDER tags.

You can choose to instantiate individual templates (TEMPLATE) or combine multiple templates in a chart (BUILDUP). In the template, enter the name of the template (NAME) and the name of the chart to be created (INSTANCE) as attributes.

Element	Attribute	Description
TEMPLATE		*Template*
	NAME	Template name
	INSTANCE	Name of the chart to be created

If you want to combine multiple templates to create a chart, you must enclose the template elements in a BUILDUP element. In this case, the name of the chart to be created should be specified in the BUILDUP element rather than in each individual template element:

Element	Attribute	Description
BUILDUP		*Grouped templates*
	INSTANCE	Name of the chart to be created
	ALIGNMENT	HOR: Horizontal alignment (side by side) VER: Vertical alignment (one below the other)
	WIDTH	Maximum width of the chart in pixels (for horizontal alignment)
	HEIGHT	Maximum height of the chart in pixels (for vertical alignment)

Any number of replacements for placeholders can be specified within a template element using the SUBST element.

Element	Attribute	Description
SUBST		*Substitution*
	NAME	Name of the placeholder
	VALUE	Value to be replaced

Charts can also be created without using templates. To do this, use the DIAGRAM element:

Element	Attribute	Description
DIAGRAM		*Chart*
	NAME	Name of the chart to be created
	WIDTH	Width of the chart to be created in pixels
	HEIGHT	Height of the chart to be created in pixels

Components on a chart are described with the COMP element:

Element	Attribute	Description
COMP		*Component*
	UID	The unique identifier of the component type
	NAME	The instance name of the component
	REF	A freely selectable reference name for the component
	CYCLE	The time slice to which this component is to be assigned (1 to 8).

The type of component is identified by the unique UID. For the component type to be instantiated, it must be in a library, which means it must be available in the "Components" task card.

You also need to specify the name (NAME) to be assigned to the component on the chart. If this name already exists when it is imported into the project, SIMIT automatically adds a consecutive number to make the name unique.

You only need to specify a reference name (REF) if the name of the component is not unique within the import file and a connection to the component is to be specified. This can happen with connectors, for example, which occur repeatedly with the same name.

The time slice to which this component is to be assigned can be specified with the CYCLE element. If this attribute is not specified, a time slice value of 2 is entered.

Links to an existing component are described with the LINK element:

Element	Attribute	Description
LINK		*Component*
	UID	The unique identifier of the component type
	NAME	The instance name of the component

The position of a component or a link is specified by the POS element and its X and Y attributes. The top left corner of the component is specified. The zero point of the chart is in the top left corner of the chart. Positions must be positive and located within the dimensions of the chart:

Element	Attribute	Description
POS		*Position*
	X	The X position of the component in pixels
	Y	The Y position of the component in pixels

A component can also be scaled and rotated with the TRANSFORM element:

Element	Attribute	Description
TRANSFORM		*Transformation (scaling, rotation)*
	SCALEX	Scaling in X direction (Default: 1).
	SCALEY	Scaling in Y direction (Default: 1).
	ROTATION	Angle of rotation in degrees by which the component is to be rotated counterclockwise. The center of rotation is the geometric center of the component.

The parameters or a component or a link can be specified with the PROP element and its NAME and VALUE attributes:

Element	Attribute	Description
PROP		*Parameter*
	NAME	Parameter name
	VALUE	Parameter value

Component connectors can be joined together. In some cases this takes place automatically if they are superimposed on the chart and have the same connection type. In other cases the

connection has to be specified in the import file. Start by specifying the name of the I/O with the PORT element:

Element	Attribute	Description
PORT		*Component connector*
	NAME	Connector name

The connection is then described by the CONNECTION element:

Element	Attribute	Description
CONNECTION		*Connection*
	TYPE	LINE: Connection using signal line IMPLICIT: Implicit connection
	SOURCE	The name of the component with which the connection is to be established. If the component has the REF attribute, its value must be used here.
	NAME	The name of the connector with which the connection is to be established.

The connection of directional signals should always be defined starting from the input, as the input can only be connected to exactly one output of another component. Therefore the connector to be specified in the connection definition is always the output of a component. As topological connections are directionless, with this type of connection it makes no difference which of the two connectors to be connected is specified as the connector in the connection definition.

The SOURCE attribute contains the name of the component or coupling to be connected. The NAME attribute contains the name of the I/O or signal. For connections of the type IMPLICIT, the name of the component must be specified in the SOURCE attribute. Reference names (REF) are not permitted for implicit connections in this case.

I/O default settings are specified with the DEFAULT element and its NAME and VALUE attributes.

Element	Attribute	Description
DEFAULT		*Default*
	NAME	Connector name
	VALUE	Default

If no default settings are specified, the values defined in the component type apply for the connectors.

In accordance with the usual XML conventions, comments are also permitted in the import file:
`<!-- Comment -->`

See also

Import charts from ZIP or XML files (Page 299)

4.3.3 Examples of XML files

You can find the examples for XML files in the Windows Start menu under "SIMIT" > "Samples" > "xml".

4.3.4 Structure of the control file "procedure.cfg" for import from ZIP file

Introduction

The ZIP archive must contain the "procedure.cfg" control file at the top level. You use this control file to define where the imported components are stored in SIMIT.

Structure of the control file "procedure.cfg"

[RETRIEVE_COMPONENTS]

- source=<ZIP file>
  Required. Specifies the ZIP file containing the components. Only unencrypted ZIP files are supported.
- area=<user | project>
  Optional. Specifies whether the components are stored in the "User components" or "Project components" when the zip file is unzipped in the "Components" task card. If you do not make a setting here, the components are stored under "Project components".
- destination=<folder name>
  Optional. Specifies the folder under "User components" or "Project components" in which the components are to be saved. If you do not make a setting here, the components are stored in the highest-level folder.

[COPY_COMPONENTS]

- source=<folder name>
  Required. Specifies the folder in the ZIP archive that contains the components in "*.simcmp" format. Subfolders are included.
- area=<user | project>
  Optional. Specifies whether the components in the "Components" task card are imported to "User components" or "Project components". If you do not make a setting here, the components are imported to "Project components".
- destination=<folder name>
  Optional. Specifies the folder under "User components" or "Project components" to which the components are to be imported. If you do not make a setting here, the components are imported to the highest-level folder.

[GENERIC_IMPORT]

- file=<XML file>
  Required. XML file that contains the import information.
- destination=<folder name>
  Optional. Specifies the folder under "Project navigation > Charts" to which the components are to be imported. If you do not make a setting here, the components are imported straight to "Charts".
- RemoveEmptyElementsWithEmptyReplacement=<True | False>
  Optional. Removes components containing empty replacements. The default is "False".

See also

Automatic modelling > Automated import (Page 980)

Import charts from ZIP or XML files (Page 299)

4.4 Bulk engineering

4.4.1 Exporting data

Introduction

You use bulk engineering to edit the parameters of a project in a table without opening individual charts or components. You can export parameters and input defaults in a spreadsheet (*.xlsx), edit the values and then import the content of the spreadsheet to the project again. For structural changes to the simulation model, use the mechanisms for automatic model generation.

Requirement

- SIMIT project is open.
- Project view is displayed.

Procedure

Follow these steps to export data:

1. Select the "Bulk engineering export" command from the "Charts" shortcut menu in the project tree.
   The command is available for the "project" and "chart" elements and for all elements below charts.
   If you run the export on elements below the "Charts" level, only the parameters and input defaults of the components from that selected level on will be exported.
2. Enter a name for the export file.

Result

The export file is generated and saved in the specified directory.

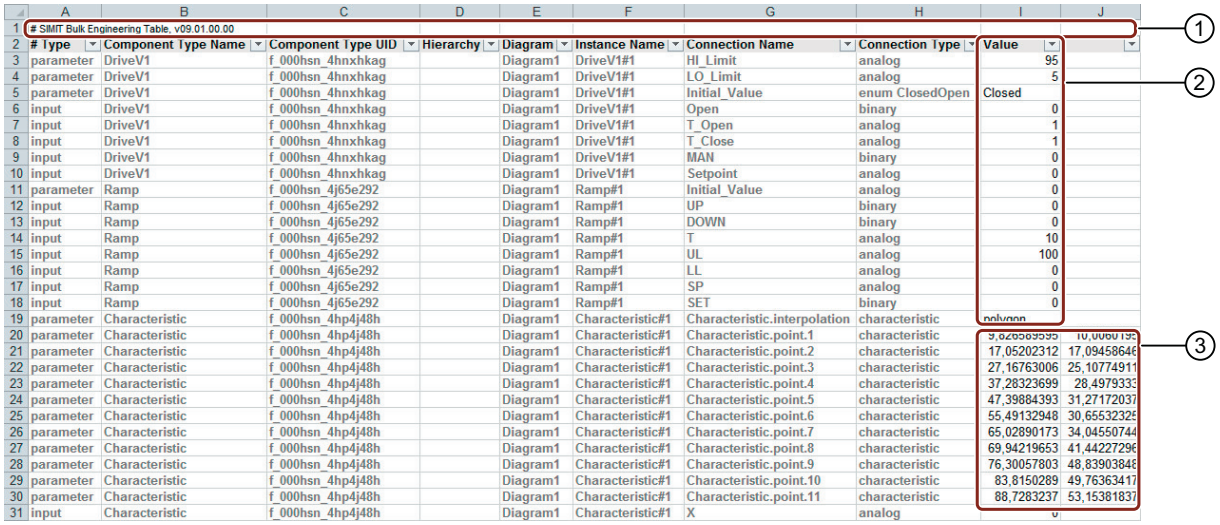
See also

Automatic model creation (Page 279)

4.4.2 Content of the export file

Structure of the export file

The figure below shows the structure of the export file:



	A	B	C	D	E	F	G	H	I	J
1	# SIMIT Bulk Engineering Table, v09.01.00.00									
2	# Type	Component Type Name	Component Type UID	Hierarchy	Diagram	Instance Name	Connection Name	Connection Type	Value	
3	parameter	DriveV1	f_000hsn_4hnxhkag		Diagram1	DriveV1#1	HI_Limit	analog	95	
4	parameter	DriveV1	f_000hsn_4hnxhkag		Diagram1	DriveV1#1	LO_Limit	analog	5	
5	parameter	DriveV1	f_000hsn_4hnxhkag		Diagram1	DriveV1#1	Initial_Value	enum ClosedOpen	Closed	
6	input	DriveV1	f_000hsn_4hnxhkag		Diagram1	DriveV1#1	Open	binary	0	
7	input	DriveV1	f_000hsn_4hnxhkag		Diagram1	DriveV1#1	T_Open	analog	1	
8	input	DriveV1	f_000hsn_4hnxhkag		Diagram1	DriveV1#1	T_Close	analog	1	
9	input	DriveV1	f_000hsn_4hnxhkag		Diagram1	DriveV1#1	MAN	binary	0	
10	input	DriveV1	f_000hsn_4hnxhkag		Diagram1	DriveV1#1	Setpoint	analog	0	
11	parameter	Ramp	f_000hsn_4j65e292		Diagram1	Ramp#1	Initial_Value	analog	0	
12	input	Ramp	f_000hsn_4j65e292		Diagram1	Ramp#1	UP	binary	0	
13	input	Ramp	f_000hsn_4j65e292		Diagram1	Ramp#1	DOWN	binary	0	
14	input	Ramp	f_000hsn_4j65e292		Diagram1	Ramp#1	T	analog	10	
15	input	Ramp	f_000hsn_4j65e292		Diagram1	Ramp#1	UL	analog	100	
16	input	Ramp	f_000hsn_4j65e292		Diagram1	Ramp#1	LL	analog	0	
17	input	Ramp	f_000hsn_4j65e292		Diagram1	Ramp#1	SP	analog	0	
18	input	Ramp	f_000hsn_4j65e292		Diagram1	Ramp#1	SET	binary	0	
19	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.interpolation	characteristic	polygon	
20	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.1	characteristic	9,826585595	10,0060195
21	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.2	characteristic	17,05202312	17,0945864
22	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.3	characteristic	27,16763006	25,1077491
23	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.4	characteristic	37,28323699	28,4979333
24	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.5	characteristic	47,39884393	31,2717203
25	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.6	characteristic	55,49132948	30,6553232
26	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.7	characteristic	65,02890173	34,0455074
27	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.8	characteristic	69,94219653	41,4422729
28	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.9	characteristic	76,30057803	48,8390384
29	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.10	characteristic	83,8150289	49,7636341
30	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.11	characteristic	88,7283237	53,1538183
31	input	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	X	analog	0	

① SIMIT ID for import.
Do not change this line.

② Parameter values and defaults of open inputs

③ Parameter pairs of characteristic curves

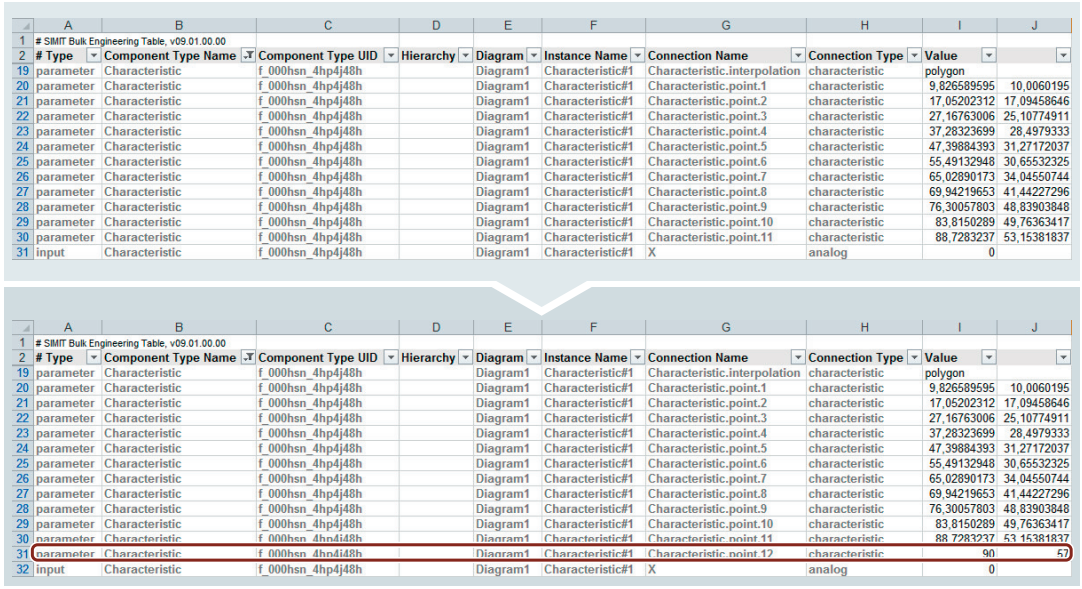
Rules for working in the export file

Observe the following rules:

- Only change values in the "Value" column.
- When changing values, check the corresponding data type in the column "Connection_Type".
- Do not change the number or order of columns.
- Do no add any lines. Exception: "Adding parameter pairs"

Adding parameter pairs

You can add parameter pairs of characteristic curves to the export file. The continuation of a range of parameter pairs is supported:



	A	B	C	D	E	F	G	H	I	J
1	# SIMIT Bulk Engineering Table, v09.01.00.00									
2	# Type	Component Type Name	Component Type UID	Hierarchy	Diagram	Instance Name	Connection Name	Connection Type	Value	
19	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.interpolation	characteristic	polygon	
20	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.1	characteristic	9,826589595	10,0060195
21	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.2	characteristic	17,05202312	17,09458646
22	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.3	characteristic	27,16763006	25,10774911
23	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.4	characteristic	37,28323699	28,4979333
24	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.5	characteristic	47,39884393	31,27172037
25	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.6	characteristic	55,49132948	30,65532325
26	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.7	characteristic	65,02890173	34,04550744
27	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.8	characteristic	69,94219653	41,44227296
28	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.9	characteristic	76,30057803	48,83903848
29	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.10	characteristic	83,8150289	49,76363417
30	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.11	characteristic	88,7283237	53,15381837
31	input	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	X	analog	0	

	A	B	C	D	E	F	G	H	I	J
1	# SIMIT Bulk Engineering Table, v09.01.00.00									
2	# Type	Component Type Name	Component Type UID	Hierarchy	Diagram	Instance Name	Connection Name	Connection Type	Value	
19	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.interpolation	characteristic	polygon	
20	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.1	characteristic	9,826589595	10,0060195
21	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.2	characteristic	17,05202312	17,09458646
22	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.3	characteristic	27,16763006	25,10774911
23	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.4	characteristic	37,28323699	28,4979333
24	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.5	characteristic	47,39884393	31,27172037
25	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.6	characteristic	55,49132948	30,65532325
26	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.7	characteristic	65,02890173	34,04550744
27	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.8	characteristic	69,94219653	41,44227296
28	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.9	characteristic	76,30057803	48,83903848
29	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.10	characteristic	83,8150289	49,76363417
30	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.11	characteristic	88,7283237	53,15381837
31	parameter	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	Characteristic.point.12	characteristic	90	57
32	input	Characteristic	f_000hsn_4hp4j48h		Diagram1	Characteristic#1	X	analog	0	

4.4.3 Importing data

Introduction

You use the "Bulk engineering import" function to import the following data:

- Data exported and edited with the "Bulk engineering export" function
- Changed data from the last simulation run. The changes are saved in a file when you exit the simulation.

Requirement

- SIMIT project is open.
- A file suitable for bulk import is available.

Procedure

Follow these steps to import data:

1. In the "Automatic modelling" menu, select the "Bulk engineering import" command. The "Bulk engineering import" dialog will open.
2. If you want to import data from the last simulation run:
   – Select the option "Apply changes from last simulation".

3. If you want to import data with the "Bulk engineering export" function:
   - Disable the option "Apply changes from last simulation".
   - Select the file from which you want to import the changed values.
4. If necessary, open the preview and select the elements that you want to import.
5. Start the import.

Result

The changed values are imported to the SIMIT project. If a parameter is outside a defined valid range, the relevant maximum or minimum value is imported. Values with errors are not imported.

See also

"Bulk engineering import" dialog box (Page 996)

Diagnostics & visualization 5

5.1 Trend and messaging editor

5.1.1 Functions of the Trend and Messaging Editor

The Trend and Messaging Editor supports the analysis of processes and signal values in the simulation with the following functions:

- **New trend**
  A trend displays the course of signals graphically and can also be printed. You can find additional information on this in section: Trends (Page 314).
- **Messages**
  The messaging system generates messages for events in the simulation. The messages can be evaluated in the message editor. You can find additional information on this in section: Message system (Page 309).
- **Archive**
  The archive allows you to record signals in a simulation run. You can find additional information on this in section: Archive (Page 312).

The functions are located in the "Monitoring" folder in the project tree.

5.1.2 Message system

5.1.2.1 Message classes

Messages provide information on the occurrence of certain events within the simulation. Depending on the triggering event they may be classified as follows:

- **Messages sent by the message component**
  A designated message component sends a message. This component is part of the standard library and can be linked with binary signals on charts. The message category and message text can be specified individually for any instance of the message component.
- **Messages sent by other components**
  These messages are defined in the component types and have a fixed message category and message text that are stored in the component type.
- **System messages**
  System messages are generated by SIMIT itself and are used to inform you about unexpected events, for example, problems when establishing a coupling connection. SIMIT system messages are always of the *SYSTEM* category.

All messages that are created after simulation starts are buffered in SIMIT and are available in the message editor. Existing messages are deleted when the simulation is restarted.

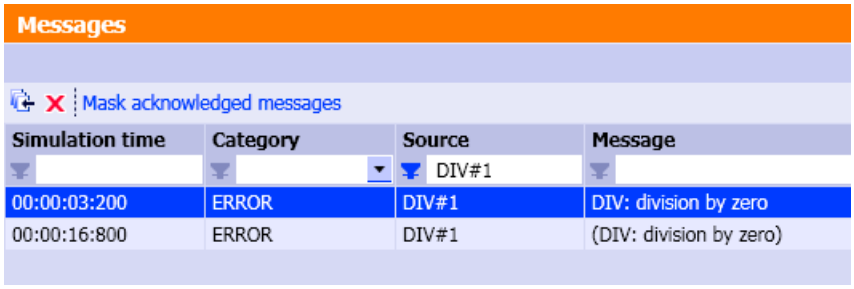
5.1.2.2 Message editor

Double-click to open the message editor under "Monitoring > Messages" in the project navigation. The messages are listed with the following information:

- Simulation time in the format "hh:mm:ss:msec"
- Category
- Name of the component that has triggered the message (source)
- Message text (message)

The message editor is available both when simulation is running and when simulation is stopped. If the simulation is running, new messages are added as soon as they are issued. After simulation is stopped, all messages generated in the preceding simulation run are listed.

The messages displayed can be filtered by a range of criteria. In the figure below, a filter for messages with the source *DIV#1* is set:



5.1.2.3 Editing messages

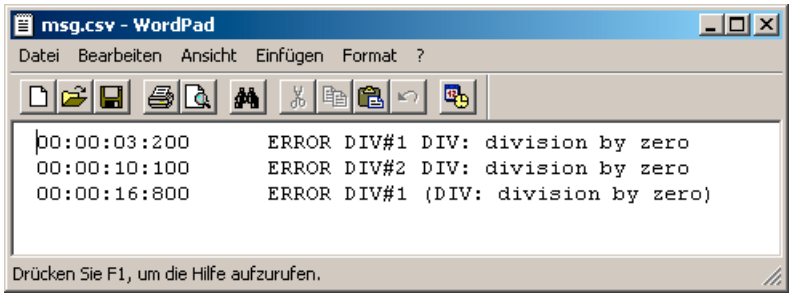
Exporting messages

Export the messages by clicking on the "" symbol in the message editor. Export is in a text file (*.txt). The text file has 4 columns:

- Simulation time
- Category
- Name of the triggering component
- Message text

The columns are tab-delimited.

The text file allows you to archive messages and edit them with a text editor.



Deleting messages

Delete all messages in the message editor by clicking on the "x" symbol.

Hiding messages

A message component generates messages that are triggered by the binary signal to which it is connected. When the signal changes from zero to one, a message is generated as an "incoming" message; when the signal changes from one to zero, an "outgoing" message is generated. "Outgoing" messages are indicated by parentheses around the message text.

A pair of messages consists of an "incoming" message and a subsequent "outgoing" message for the same event. You can use the "Hide message pair" button to hide and show message pairs. If you hide message pairs, the only messages that remain visible are those that have "come in" but have not "gone out" yet.

5.1.2.4 Message – the message component

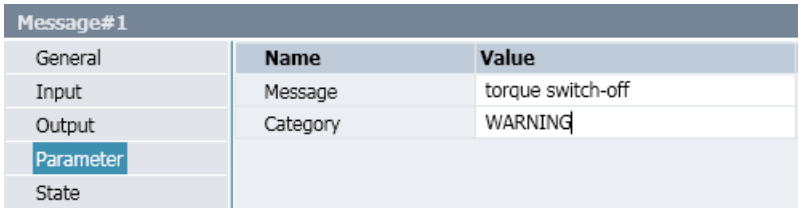
Symbol



Function

The *Message* component type is to be found in the standard library *Misc* folder. A component of this type will generate a message when the binary value at its input changes. In the case of an "outgoing" message, the message text that is output by the component is placed in parentheses.

You can specify the category and message text as parameters in the component property view.



Note

Messages in the SYSTEM category are system messages from SIMIT. Messages in the ERROR category are messages from components in the basic library. When choosing category names, avoid using SYSTEM and ERROR. When filtering messages by category, this will allow to see just the messages from your own categories.

5.1.2.5 Component-specific messages

Some components from the basic library generate messages to notify you of invalid operations (for example division by 0 or invalid parameter assignment).

For example, a component of the type SQRT outputs the message "SQRT: invalid argument" if the input value has become negative.

Messages from basic library components are always assigned to the ERROR message category, and their message text is output in the format "Component type: Error message". All messages are created as "incoming" or "outgoing".

5.1.2.6 Messages in the status bar

The current message is shown on the right of the SIMIT status bar along with its simulation time and the message text:



5.1.2.7 Limitations of the message system

The message system can process up to 1000 messages per second. If this value is exceeded, messages are lost.
If this occurs, the following message is displayed:

00:00:02:600	SYSTEM	ControlSystem Warning	TME : Message system overload

A maximum of 4096 messages can be saved. If this limit is reached, the following message appears:

00:02:58:600	SYSTEM	TME	Message buffer full

5.1.3 Archive

The archive records the signals of a simulation run, which means from simulation start until simulation stop. The next time simulation is run, i.e. the next time simulation is started, the archive is cleared and the signals to be archived are once again recorded until the simulation stops.

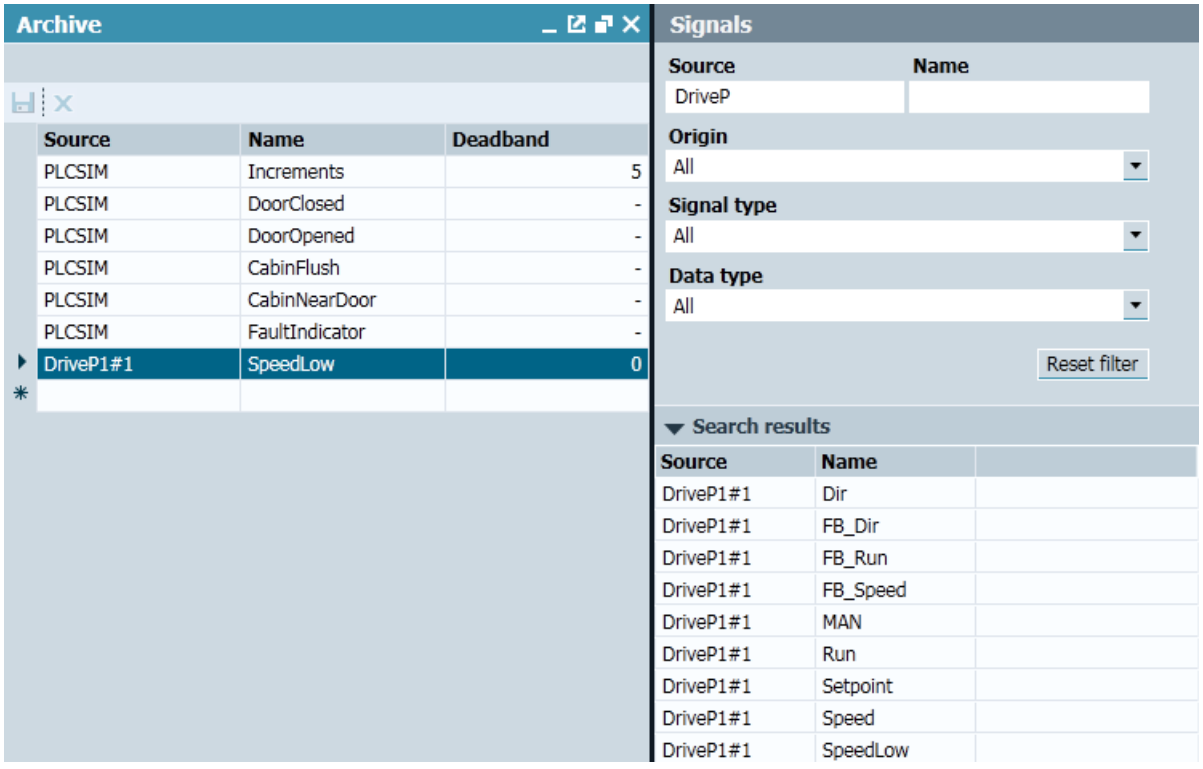
The following signals are recorded in the archive:

- Component input and output signals
- Component states
- Coupling input and output signals

Archived signals can be visualized as trends.

Configuring the archive

Open the archive by double-clicking on the archive under "Monitoring > Archive" in the project navigation. The archive editor opens and the signals currently archived are displayed. Manually enter any other signals or drag them to the table from the "Signals" task card.



A deadband is defined in the archive editor to limit the quantity of data generated by recording the signals. The deadband specifies how much a signal value must change before the value is recorded. A signal value is only recorded if the value differs from the last recorded value by more than the deadband. A deadband can only be specified for analog and integer signals. The value of a binary signal is recorded every time it changes.

Note

If you enter a deadband of "0", a value is recorded in each cycle for analog and integer signals that change continuously.

A maximum of 128 signals can be entered in the archive. If this limit is exceeded, an error message is displayed.

Memory is reserved for each signal. With a basic cycle of 100 ms and a deadband of "0", a signal is therefore recorded for about 2.5 hours. Less frequent signal changes result in longer recording times.

5.1.4 Trends

5.1.4.1 Signal with trends

Trends allow you to graphically display the change of signal values over time. You can display the following signals in trends:

- Component input and output signals
- Component states
- Coupling input and output signals

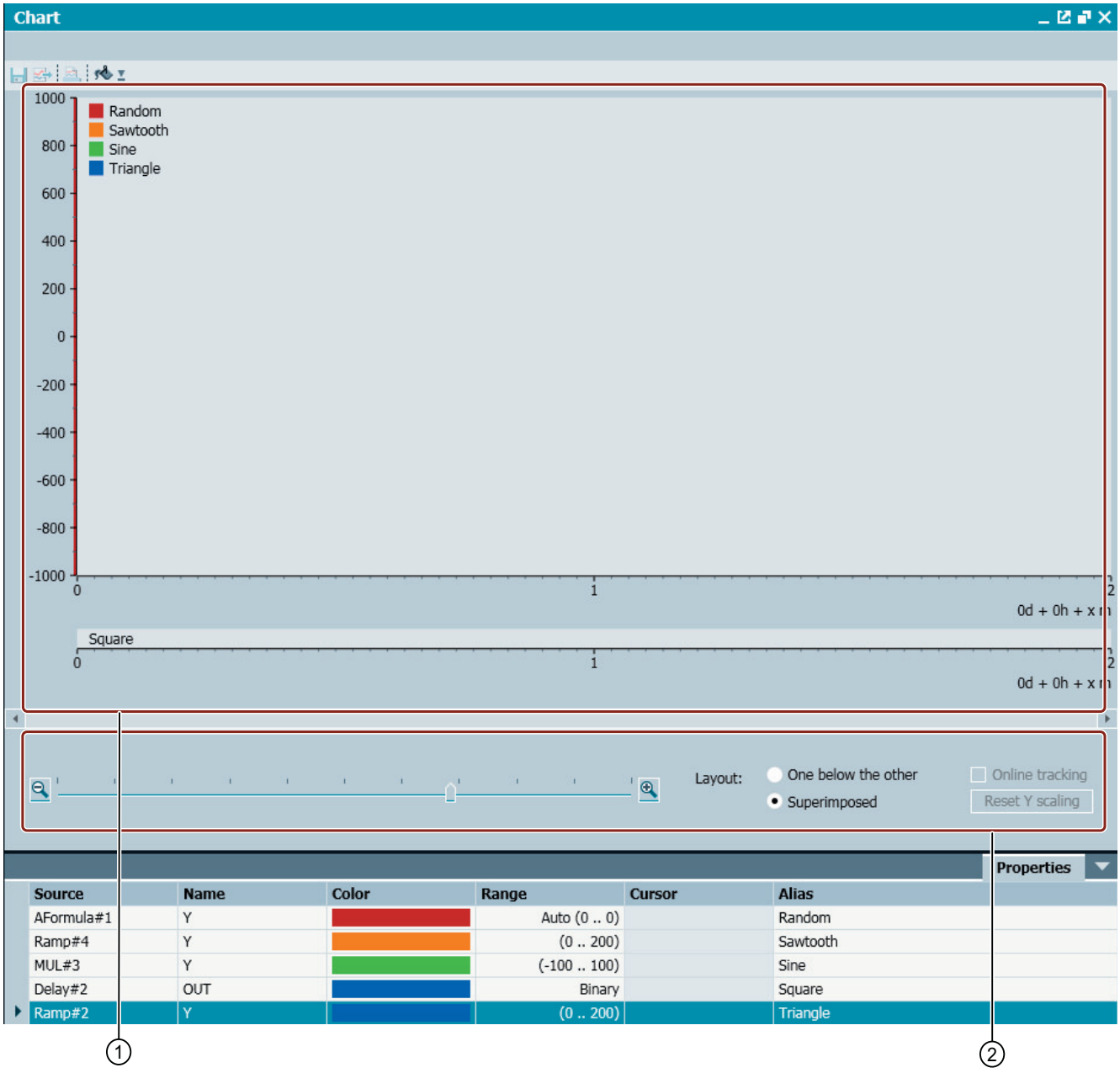
The signals may be either archived or not archived. Archived signals can be displayed with their complete trend history as recorded in the archive.

Signals that are not archived are only displayed as long as the trend is open. The trend reserves memory for each signal that is not archived. With a basic cycle of 100 ms, a signal can therefore be recorded for about 35 minutes if its value changes in each cycle. Less frequent signal changes result in longer recording times.

5.1.4.2 Adding and configuring trends

Trends are stored in the "Monitoring" folder in the project navigation. Double-click on "New trend" to create a new trend. You can create any number of trends in your project. You can create and configure trends even if you have already started simulation.

Trends are opened in the trend editor within the work area. The trends of the signals are displayed in the upper section of the trend window, and settings for the trend are made in the lower section.



① Curve characteristics

② Settings

Configuring trend signals

In the property view of the trend editor, you list the signals that are to be displayed in the trend. You can manually enter signals into the table with their source and name or drag the signals to the table from the "Signals" task card.

Source	Name	Color	Range	Cursor	Alias
AFormula#1	Y		Auto (0 .. 0)		Random
Ramp#4	Y		(0 .. 200)		Sawtooth
MUL#3	Y		(-100 .. 100)		Sine
Delay#2	OUT		Binary		Square
Ramp#2	Y		(0 .. 200)		Triangle

A trend can contain up to a maximum of 16 signals.

Assigning parameters to signals within a trend

You may specify the following parameters for each signal:

- The color in which the signal is to be displayed
- The value range of the signal
- An alias for the signal

Analog and integer signals are displayed within a certain value range. The default setting is an automatic mode "Auto (0..0)" in which the value range is continuously adapted to the values actually occurring. This mode saves you specifying an individual value range, and ensures that all signal trends are displayed in maximum resolution.

Note

It may be more difficult to compare multiple different signals in automatic mode because the scaling continuously changes.

To specify fixed ranges, you can also enter the start value and end value separated by two dots, for example: "0..2000".

If you want to label a signal in the trend with a name that differs from the signal name (which consists of the source and name), you can assign an alias to the signal. This alias is used to label the trend.

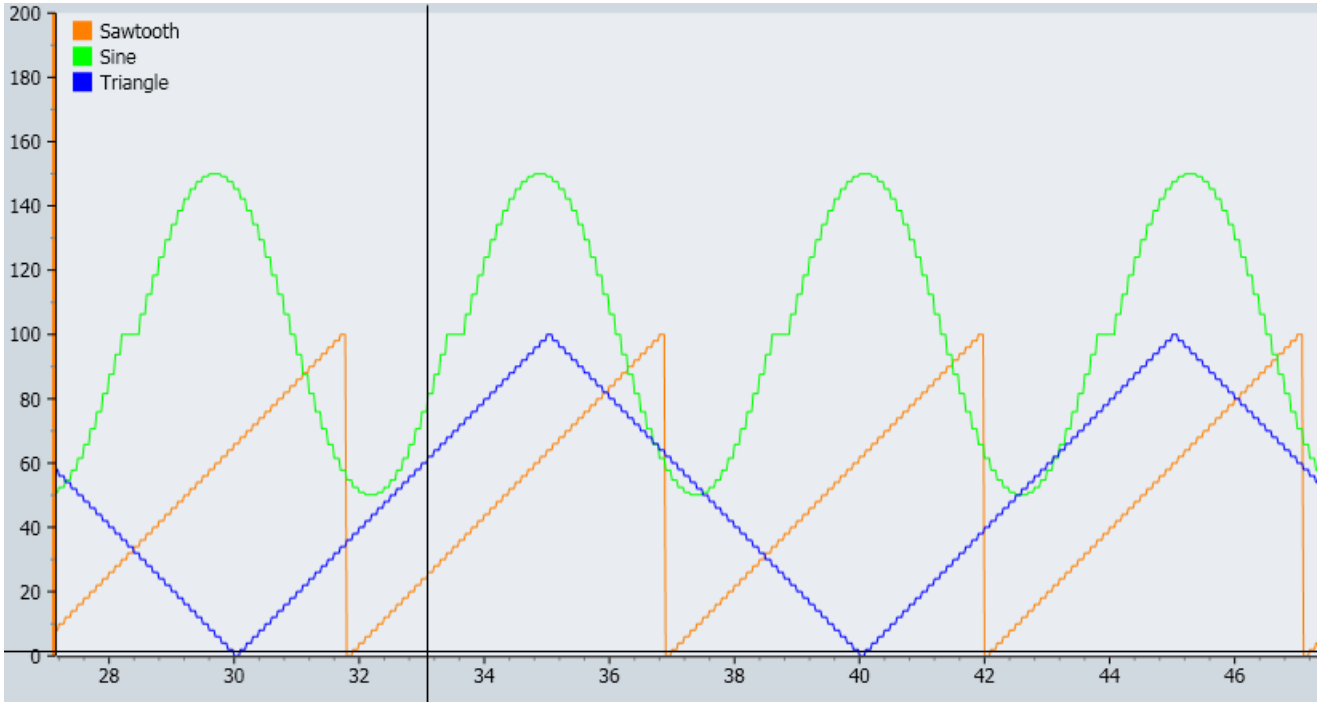
Note

You can create any number of trends in a project, but the total number of signals contained in all open trends must not exceed 128.

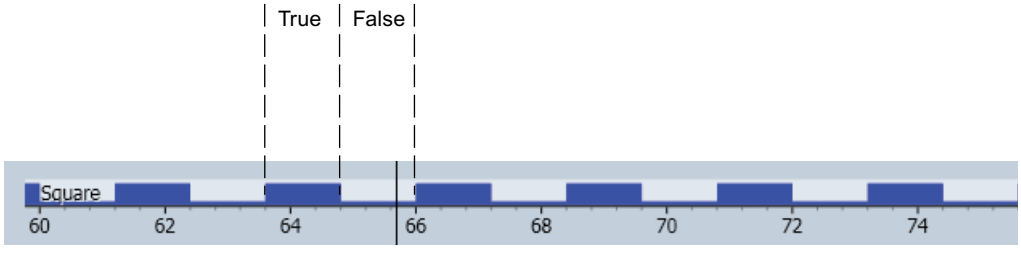
5.1.4.3 Displaying trends

Displaying trends by data type

Analog and integer signal trends are displayed in charts. The horizontal axis (X-axis) shows the simulation time and the vertical axis (Y-axis) the signal values.



Binary signals are displayed below the analog and integer signal trends. As binary signals can only have the values zero (false) and one (true), a bar graph is used to show a binary signal over the simulation time. A line represents a value of zero (false) and a bar a value of one (true), as shown in the figure below:



Displaying archived signals

Archived signals are shown with the archived signal values. As signals in the archive start being recorded when the simulation is started, the trend for archived signals always covers all times since the last time the simulation was started, provided the memory limit has not been exceeded.

Signals that are not archived are only plotted while the trend is open. This means the trend of such signals is only shown from the time the trend was opened. If you stop a simulation in progress, you will see only the signal trend of non-archived signals in trends that were already open while the simulation was running. When a trend is closed, all of the recorded signals are cleared. This means that the signal trends of non-archived signals can no longer be displayed when the trend is reopened.

While the simulation is running the signal properties also display the current signal value.

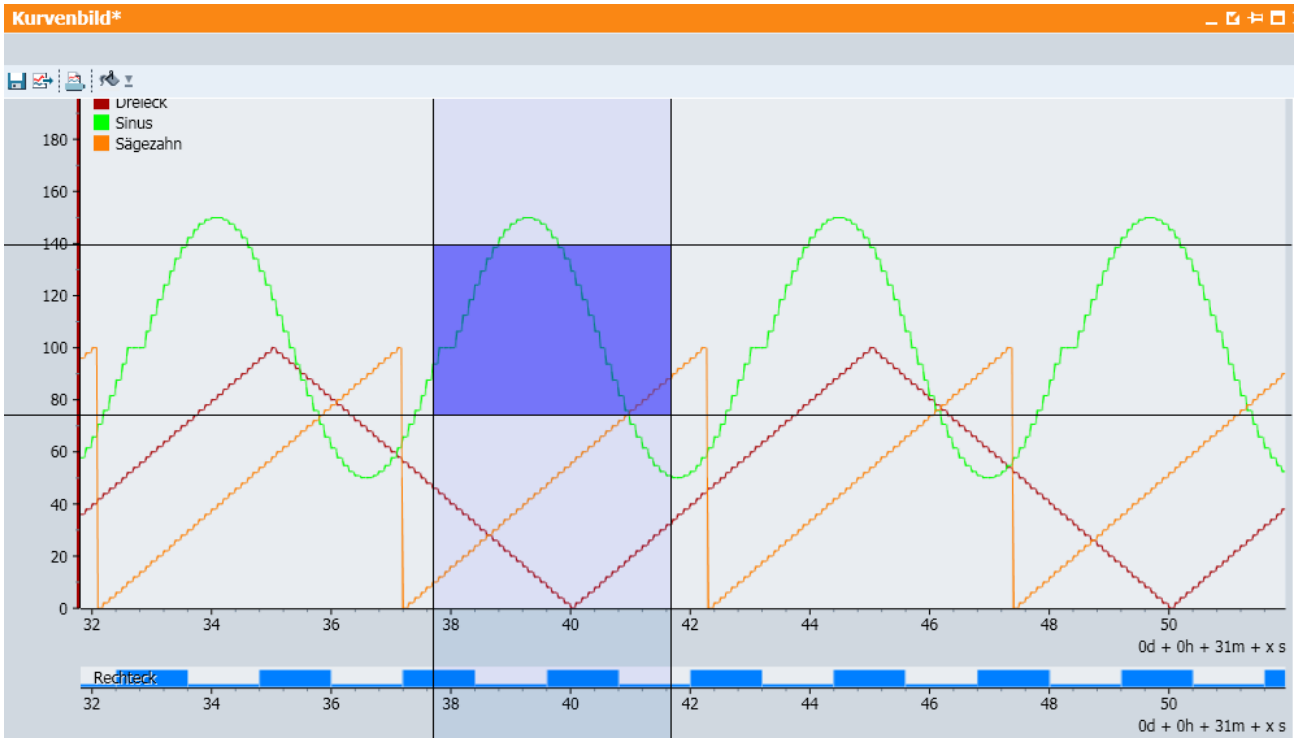
Properties

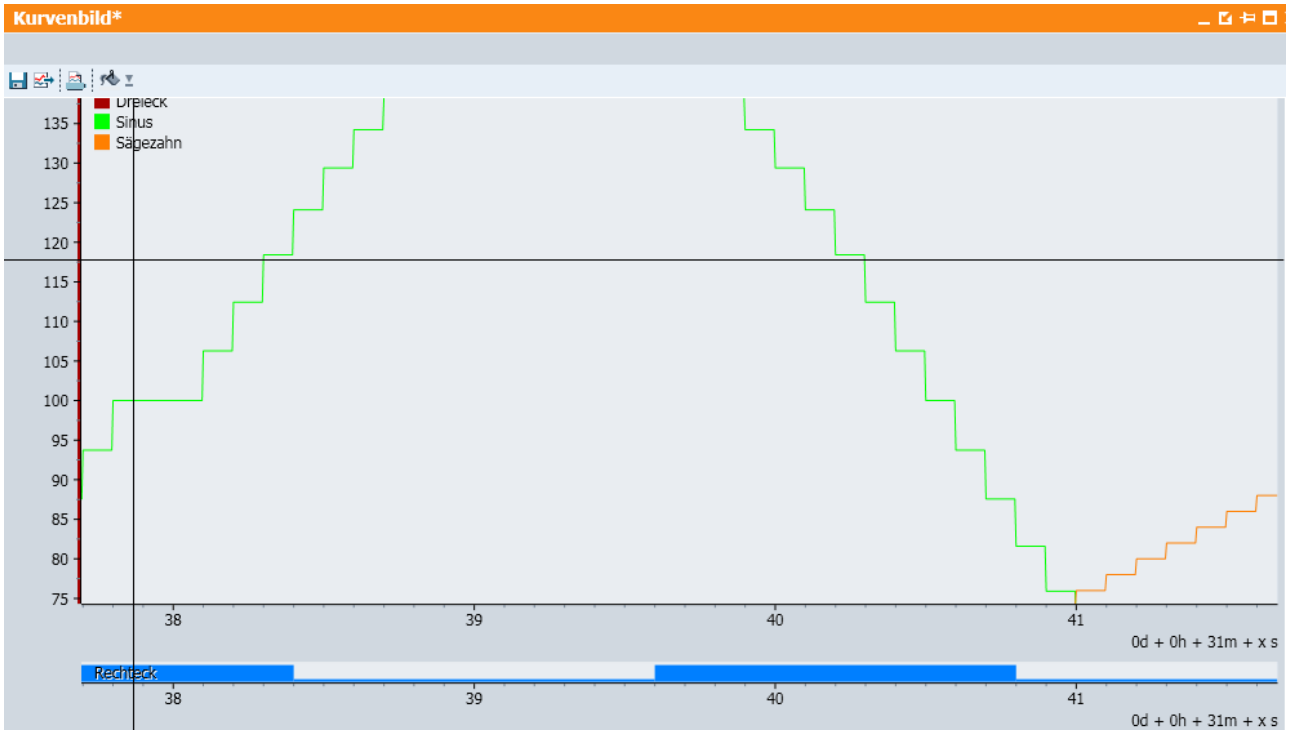
Source	Name	Color	Range	Deadband	Current	Alias	Archived
Ramp#4	Y		(0 .. 200)	0	96.0	Sawtooth	☑
MUL#3	Y		(-100 .. 100)	0	45.241352623301	Sine	☑
Delay#2	OUT		Binary	-	True	Square	☐
Ramp#2	Y		(0 .. 200)	0	74.0	Triangle	☐
*							

The deadband of archived signals is taken into account in the display of signals in the trend. The signal property view displays the dead band values while the simulation is running. For non-archived signals, all values that are calculated in the simulation are recorded in trends.

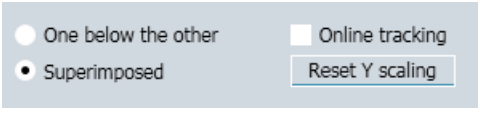
Zooming in on sections of the trend

When you turn off continuous updating of the trend and move the cursor into the trend chart, a crosshair is shown. To zoom in on a specific area, select the area with the left mouse button pressed. The selected area is then zoomed in the trend chart.





Use the "Reset Y scaling" button to reset the Y axis to the defined ranges.



You can find additional information in section: Trend settings (Page 319).

Trend settings

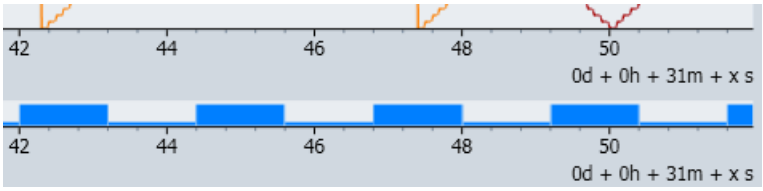
Time axis

The scale of the time axis can be set with a slider:



The time at which the visualization starts is shown on the right hand side, below the time axis. Add the corresponding number of days (d), hours (h), minutes (m) or seconds (s) according to the measured values on the X axis to this start time.

The figure below shows a section of the time axis ranging from 31 minutes and 42 seconds to 31 minutes and 50 seconds.



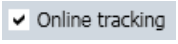
Note

Refreshing the time display

The time axis moves from right to left and shows the seconds. The measuring point for the minutes is the left-hand edge of the display. As soon as a value of 60 [s] passes the measuring point, the minute counter is incremented by 1 and the scale values set back 60.

Cyclic update

When you open a trend and start the simulation, you will see the trend values in a constantly updated sequence. The trends track the signals as they change. In order to obtain a static view of the current trends, switch off the updating. To do so, deselect the update option:



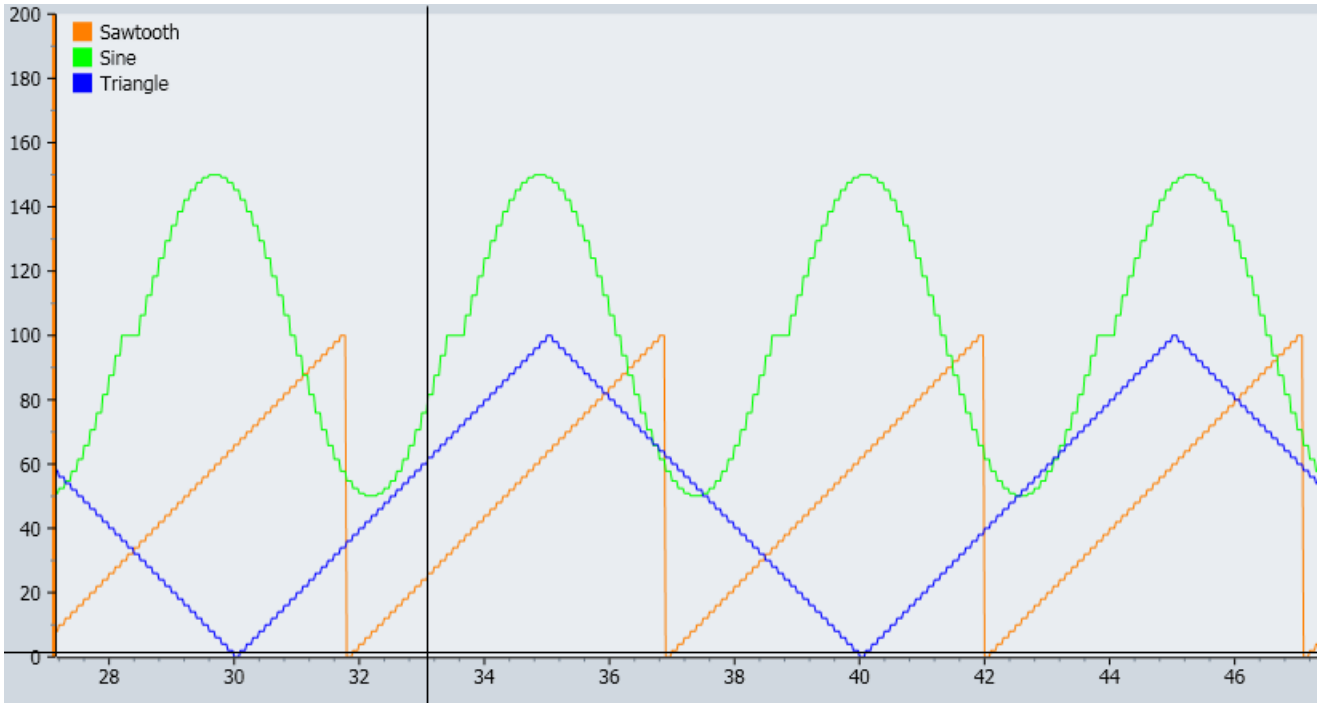
If there is no simulation running, you see a static view of the recorded signals.

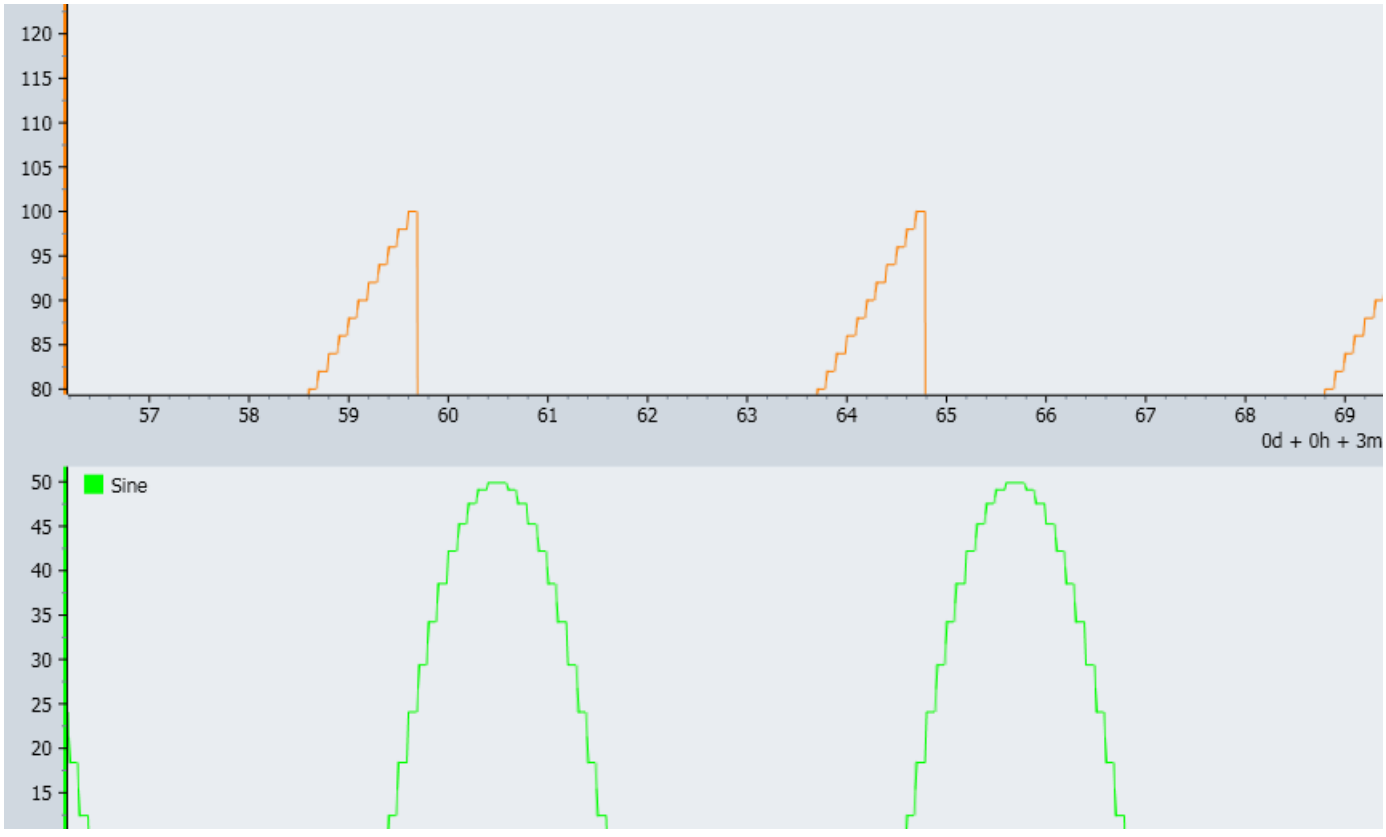
Displaying trends in superimposed mode or one below the other

You can display all analog and integer signals of a trend *superimposed* in one single chart, or all signals *one below the other* in separate charts.

In superimposed display mode, an appropriately labeled Y axis can only be shown for one of the signals. You can specify which Y axis to display by clicking on the signal name. The Y-axis is highlighted in the signal color.

The following two figures show a graph with superimposed signals (top figure) and a graph with signals one below the other (bottom figure):





The trends for binary signals are always displayed in separate trends, one below the other, with a single common time axis.

5.1.4.4 Trend editor

Features of the trend editor

You can use the trend editor to run the following functions:

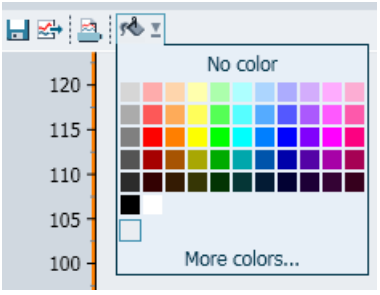
- Adapt background color of trends
- Print trends
- Export trends

The relevant commands can be found in the toolbar of the trend editor:



Background color of trends

You can freely choose the background color of a trend. To do this, use the color selection in the toolbar of the trend editor:



Printing trends

Proceed as follows to print a trend:

- Click on the "" symbol in the trend editor
  The "Print" dialog box opens.
- In the dialog box, select one or more signals to be printed and then select "one above the other" or "overlapping" signals under "Display".
  You can find the other functions of the dialog box in section: "Print" dialog box (Page 998).

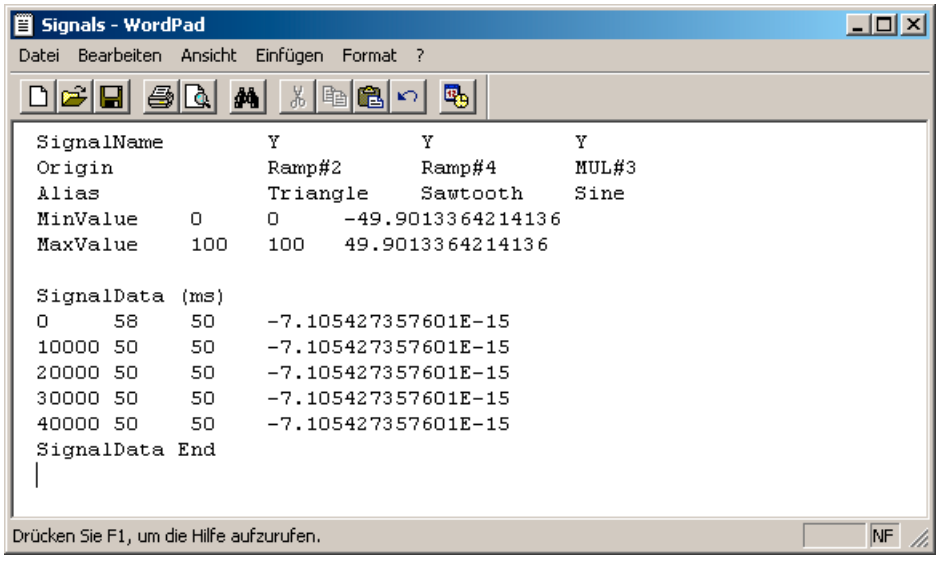
Exporting trends

Proceed as follows to export signals recorded in a trend to a text file (*.txt):

1. Click on the "" symbol in the trend editor
   The "Export" dialog box opens.
2. Select the signals you want to export.
3. Give the export file a name under "File".

4. Click the "..." button to select a memory location for the export file.
5. Under "Resolution", select a time interval for the signals. The possible settings are 100 ms, 1 s, 10 s and 60 s.

The generated file will contain the points in time and the values for each signal, with one column per signal. The individual columns are separated with the tabulator character. If there are several signal values for an analog or integer signal within a time period, the arithmetic mean between the minimum and maximum values that occurred is used.



5.2 Find & replace

The "Find & replace" function is available in the project navigation.

Double-click to open the function in the work area.

This function is used to search for elements in your project and make replacements.

You can search for the references of a signal in project elements with the "" symbol in the signal properties of components and controls.

You can find more information on the search function in: Find (Page 324).

You can find more information on the replace function in: Replace (Page 326).

Note

Find & replace in texts

You open the "Find" or "Replace" dialog box with the shortcuts <Ctrl+F> and <Ctrl+H>.

See also

"Find & replace" dialog box (Page 997)

5.2.1 Find

5.2.1.1 Finding with the Find & replace editor

The Find & replace editor opens in the work area. The "Components", "Macros" and "Signals" task cards are available in this editor.

You can search for the following elements:

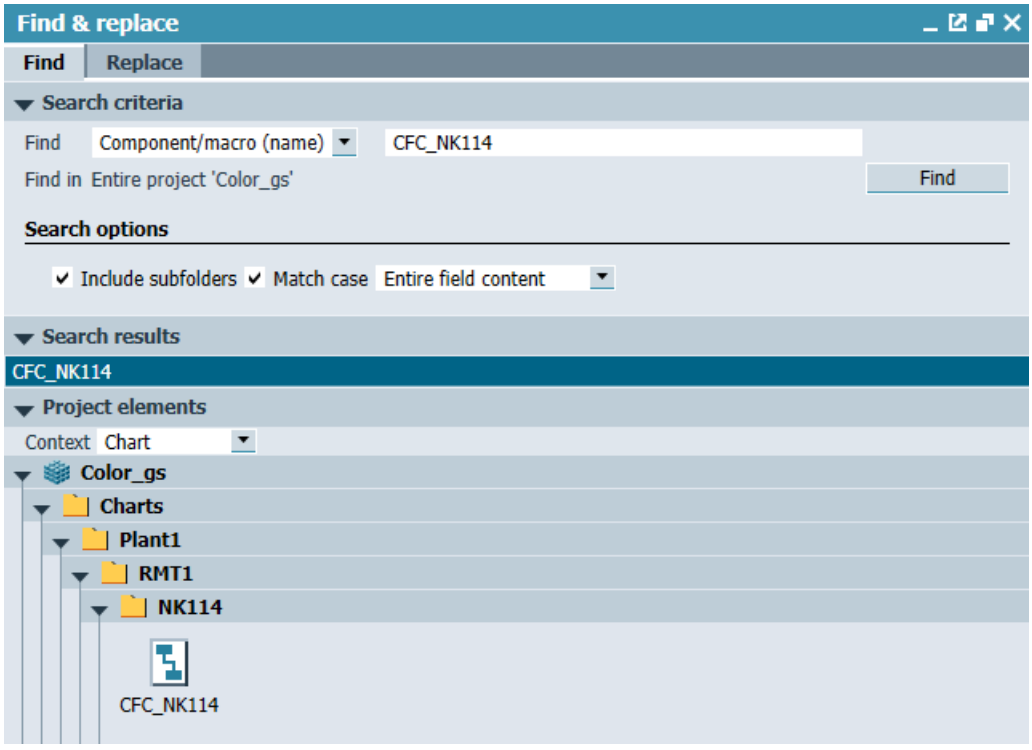
- Signals
- Diagram connectors
- Coupling connectors
- Names of components and macro components
- Components and macro components by type
- Graphic text

Note

Only saved data is searched; charts and couplings must therefore be saved before you search.

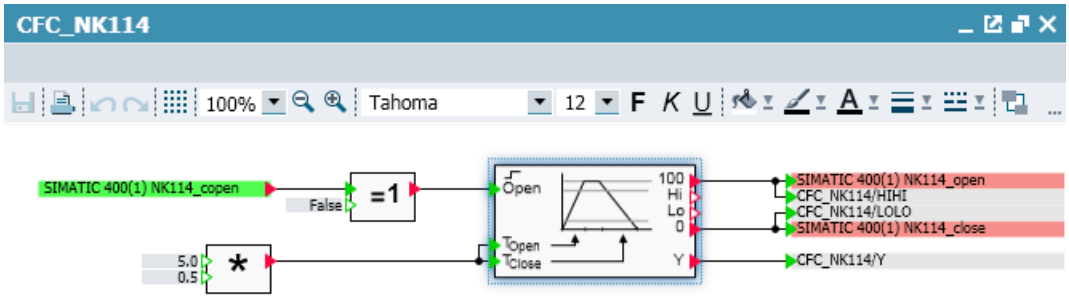
Select an element and click the "Find" button to launch the search.
You may be able to define additional search criteria and search options depending on the element selected, for example a signal name for a signal search.

In the following figure, the system searched for components and macro components that contain "CFC_NK114" in their name. The search in this case covers the entire project including all subfolders, and is case-sensitive. You can view the results under "Search results": The CFC_NK114 component.



Select a component or macro component under "Search results" to display the diagrams and links that contain the component or macro component you are looking for under "Project elements".

In the figure above, when you select "CFC_NK114" as the search result, the "CFC_NK114" diagram is displayed under "Project elements". If you open the "CFC_NK114" diagram by double-clicking, you can see that the component you are looking for is highlighted with a blue frame.



If a search result corresponds to a single chart only, you can open the chart by double-clicking on the search result.

To search for a specific signal, you can simply drag-and-drop it from the "Signals" task card to either of the two text boxes.

When searching for a component type, enter its name and/or ID. Similarly, when searching for a specific component type in your project, you can drag-and-drop a component type from the "Components" task card to the input field.

To search for components, diagram connectors and coupling connectors, enter their name. With the search option "Output connectors only", you can hide all search results of input connectors when searching for global connectors and thus navigate directly to the respective output connector.

5.2.1.2 Searching using the properties of signals and connectors

You can also search for signals and connectors in a chart using the property view.

For example, if you want to search a chart for input connectors that are connected to the SPEED output connector, open the property view and click the "Find" button ( ).

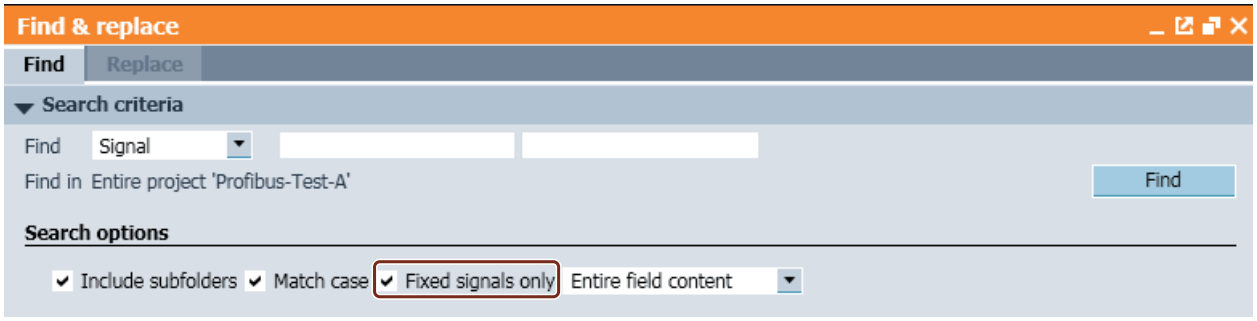


The Find & replace editor will then open with the search results.

Search results display and navigation are then as detailed in: Finding with the Find & replace editor (Page 324).

5.2.1.3 Searching for fixed signals

To search for fixed signals in the project, select the search option "Fixed signals only". This gives you an overview of what signals are currently fixed at any given time.



As signals can only be fixed when simulation is running, this search option is only available after the start of simulation.

5.2.2 Replace

5.2.2.1 Replacing with the Find & Replace editor

To replace search results in the editor, click on the "Replace" tab. You can replace

- Signals
- Diagram connectors

- Coupling connectors
- Names of components and macro components
- Components and macro components by type
- Graphic text

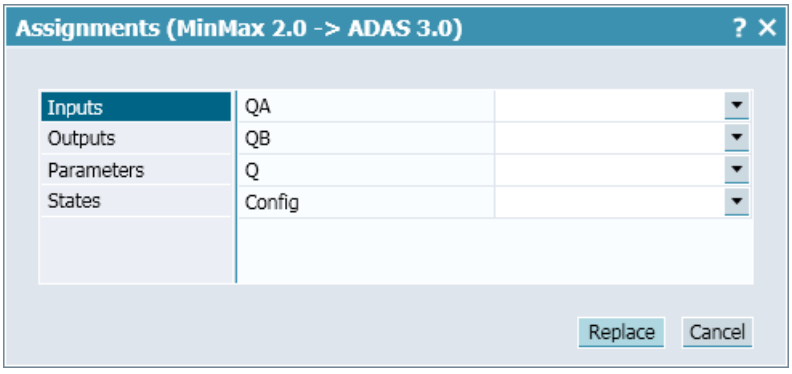
Note

Replace only works with saved data. Charts and coupling must therefore be saved before the replace function is launched.

- Enter the search criteria and options. Complete the "Replace with" text boxes and then click "Find".
- Select the results that you want to replace in the search results. In line with the search options set, you see the results of the replacement for each search result. As with the search, all charts in which the signals for which you searched were found are shown under "Project elements".

In your project, you can also replace components or macro components of a specific type by a component or macro component of another type. The type and their UID are applied.

- Click the "Replace" button.
  The "Assignments" dialog opens.



- The dialog shows the assignments of the inputs and outputs for both elements. The assignment can be changed here.
- Click the "Replace" button to apply the assignments and launch the replace function.

5.2.2.2 Refresh

Refreshing components

The search criterion "Refresh" searches by component type for all components in the defined search range that have the same name and belong to the same library family but were created (saved) at a later time.
This updates the components in your simulation project in one operation. The search for current component types is run in all component libraries in the "Components" task card in the following order: "Project components", "User components" and "Basic components". The most recently created component type is then suggested as the current type for replacement in the search result.

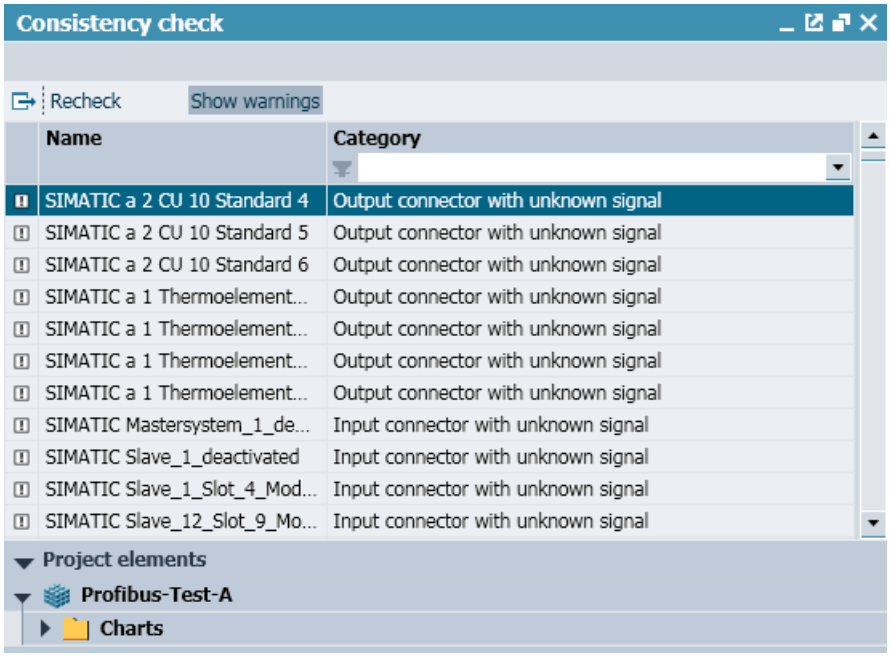
"Refresh" works in the same way for all macro components in the defined search range; the system search for current macro components in the libraries of the "Macros" task card. If all components and macro components in the selected search range are already up to date, no results are returned.

The "Refresh" option is also available if you want to update components by type in macro components. Start the update via the shortcut menu for the macro component.

5.3 Consistency check

The consistency check is always launched automatically when the simulation is started. Inconsistent projects cannot be started.

You can start the consistency check manually by double-clicking on the "Consistency check" tree node in the project navigation. If a project is consistent, a dialog appears indicating that there are no inconsistencies.



Inconsistencies are indicated by being preceded with the red symbol "". Warnings are indicated with the blue symbol "". Using a command, you may switch the display on or off for warnings.

The black symbol "" is only a warning and indicates that the simulation model may not yet be finished. A warning will not keep you from launching the simulation. The red symbol "" indicates an error that needs to be fixed before the simulation can be launched.

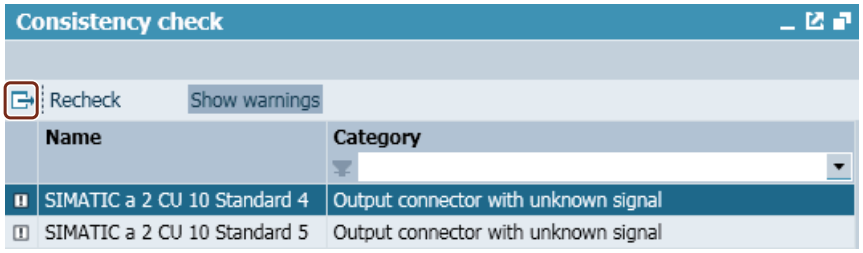
To fix an error or a warning select its entry in the list of results. The Project elements section shows the charts in which the error or warning must be resolved. From this view you can then

open the charts to fix the problem. After resolving the inconsistencies, you may recheck the project using the "Recheck" button.

Note

The consistency check only works with stored data. Data, charts and couplings must therefore be saved beforehand.

The results of the consistency check can also be exported in a tab-separated text file. To do so, activate the button in the consistency check editor as shown in the figure below.



5.4 Analysis functions

5.4.1 Basics of the analysis function

Introduction

The SIMIT analysis function generates a statistical analysis from the data of a SIMIT project. The statistical analysis is saved as an "*.xslx" analysis file. The following project data is included in analysis:

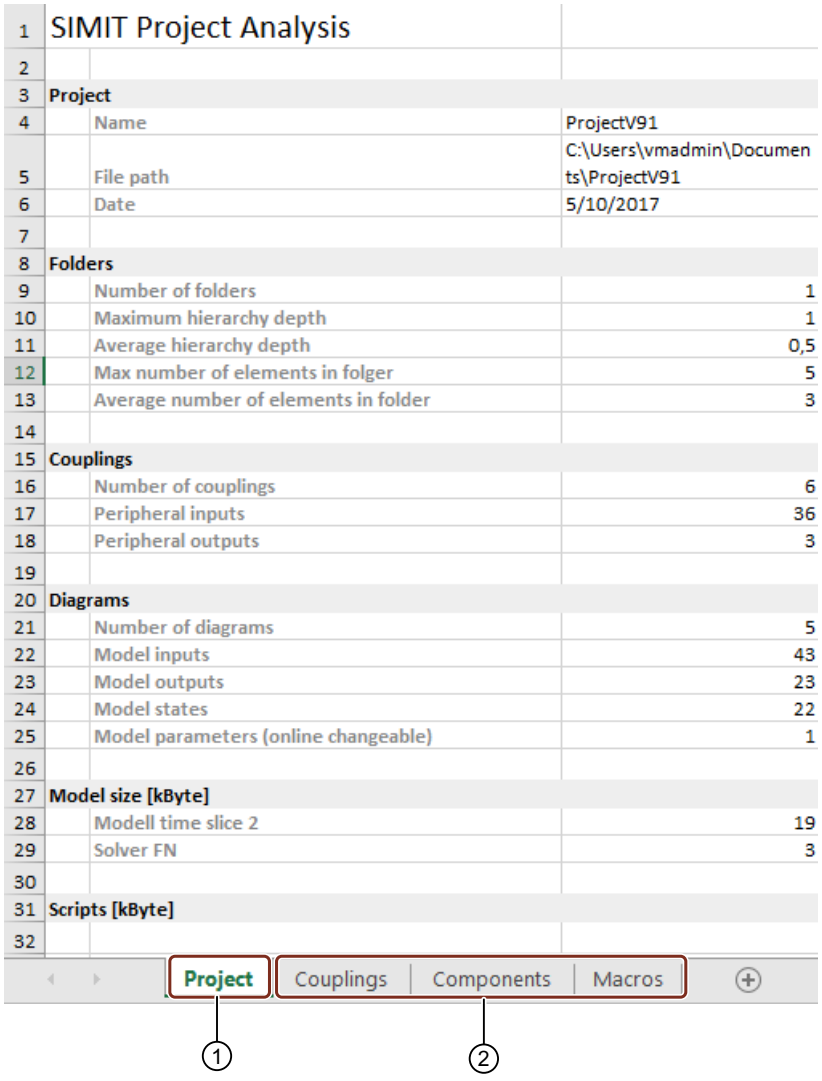
- Couplings
- Components
- Macros

Configuration limits are also calculated for the following objects:

- Folder
- Couplings
- Charts
- Model size (compiled)
- Scripts

Structure of the analysis file

The figure below shows the structure of the analysis file:



1	SIMIT Project Analysis	
2		
3	**Project**	
4	Name	ProjectV91
5	File path	C:\Users\vmadmin\Documents\ProjectV91
6	Date	5/10/2017
7		
8	**Folders**	
9	Number of folders	1
10	Maximum hierarchy depth	1
11	Average hierarchy depth	0,5
12	Max number of elements in folger	5
13	Average number of elements in folder	3
14		
15	**Couplings**	
16	Number of couplings	6
17	Peripheral inputs	36
18	Peripheral outputs	3
19		
20	**Diagrams**	
21	Number of diagrams	5
22	Model inputs	43
23	Model outputs	23
24	Model states	22
25	Model parameters (online changeable)	1
26		
27	**Model size [kByte]**	
28	Modell time slice 2	19
29	Solver FN	3
30		
31	**Scripts [kByte]**	
32		

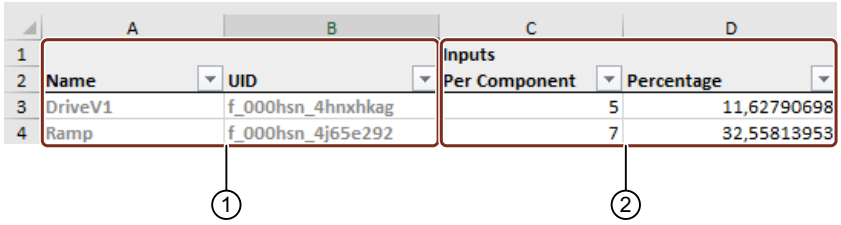
Project | Couplings | Components | Macros

① ②

① Configuration limits
② Statistical data

Content of statistical data

The figure below shows the basis structure of the statistical data, using the example of "components".



	A	B	C	D
1			Inputs	
2	Name	UID	Per Component	Percentage
3	DriveV1	f_000hsn_4hnxhkag	5	11,62790698
4	Ramp	f_000hsn_4j65e292	7	32,55813953

① ②

① The types used in the project are listed.

② The percentage is calculated on the basis of the total number.

5.4.2 Generating analyses

Requirement

- Project is open in SIMIT.
- Project view is displayed.

Procedure

1. Select the "Analysis" command from the shortcut menu in the project tree.
2. Enter a name for the analysis file.
   If Excel is installed on the PC, the analysis file will immediately open in Excel.

Result

The analysis file is generated and saved in the specified directory.

Scripts

6

6.1 How scripts work

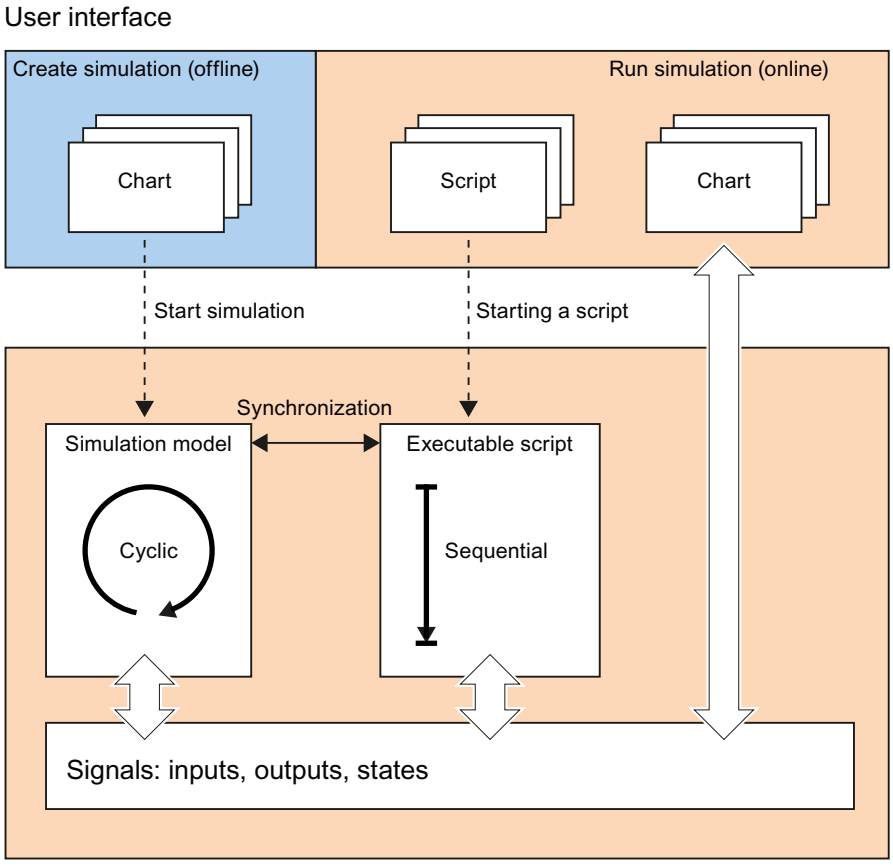
A simulation is made functional in SIMIT by defining the graphical interconnection and parameter assignment of components. When the simulation is started, an executable simulation model is generated from all charts in the simulation project and that model is then processed cyclically. The SIMIT user interface gives you access to all input and output signals, statuses and component parameters that can be changed online during ongoing simulation.

The Automatic Control Interface function module from SIMIT provides you with the additional option of using scripts to access simulation variables at specific times in ongoing simulations. You are then able to automate manual interventions through the user interface, monitor processes within the simulation, and create logs from output information.

A script is a sequence of instructions that are processed in consecutive order and without jumps from beginning to end. Time slices are used to synchronize the script and the cyclically executed simulation model. Within each time slice, the script is executed before the simulation model contained in this time slice.

You can stop a script to wait for specific events in the simulation model or for specific points in time. You can also use instructions in the script to influence the control system of the simulation model, such as creating and loading snapshots.

The schematic figure below shows how scripts are incorporated into the architecture of SIMIT:



6.2 Handling of scripts

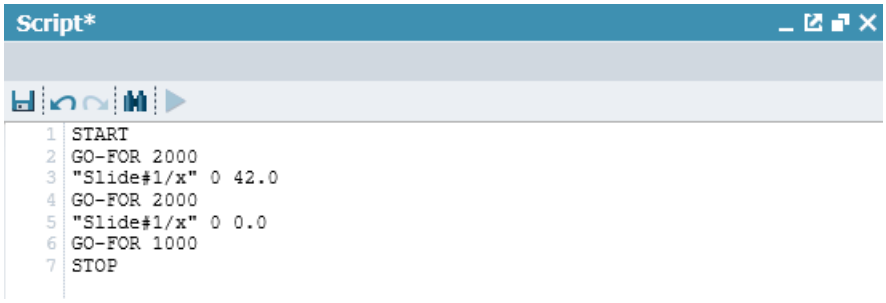
6.2.1 Creating a script

You can both create and execute scripts in your simulation project. The script editor can be used to edit scripts.

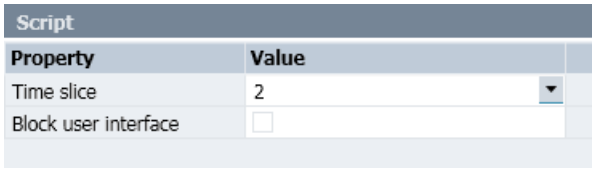
Scripts are elements of a SIMIT project and are managed in the "Scripting" folder of the project tree. You can create and edit scripts whenever SIMIT is open – both when the simulation has not yet been started (offline) and when it has been started (online). Double-click on "New script" in the "Scripting" folder to create a new script.

To edit a script, open the script editor from the shortcut menu or double-click on the script in the navigation window.

The editor that then opens lists the line number in a column to the left of the pane in order to facilitate the allocation of error messages.



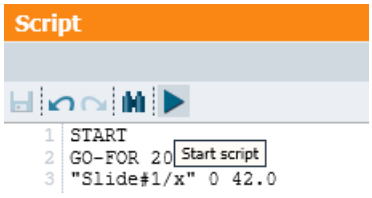
Enter the time slice in which the script is to be executed in the Property view of the script editor. If you include other scripts with an instruction, the time slice specified in the scripts included is irrelevant; the applicable time slice is the one specified in the script that has been started.



If you enable the "Block interface" option, a dialog appears when the script is started that prevents access over the interface to the simulation in progress. Close this dialog to cancel the script.

6.2.2 Executing a script

A script can only be executed when a simulation is running. You can start a script either through its shortcut menu in the project tree or by using the toolbar in the script editor.



A flashing { } symbol in the status bar indicates that a script has been started. The symbol for simulation in progress is also displayed.

You can stop a script at any time using the shortcut menu for the "Scripting" folder.

Note

Scripts can only be started in asynchronous mode.

Operating modes (Page 39).

Only one script can be run at a time. Execution of a script is stopped once all instructions in the script have been completed, when it is stopped by a command or when the simulation is closed or reinitialized in the interface.

After a script is started, the syntax is checked and the script is compiled into an executable instruction list. Any error discovered in the script is signaled and the script is not started.

Problems can also arise that only become apparent once the script is being executed. Should such problems occur, a message appears in the status bar and the script is stopped.

6.2.3 External scripts

Note

We recommend that you only create and edit scripts using the SIMIT script editor, as this will ensure that they remain in the simulation project. This is because only scripts located in the SIMIT project tree can be executed and archived with the project.

However, if you wish to create scripts externally using a different editor, note the following:

Scripts are text files with the standard Windows character encoding (code page 1252). Script files must have the file extension ".script". The script properties that are listed in the property view of the script editor must be set as follows in the first two lines of the script file in line with the information in the "Keywords for script properties" table:
`Keyword = Value`

Example:
```
META_BLOCKGUI = False
META_CYCLE = 2
```

Table 6-1 Keywords for the properties of scripts

Keyword	Meaning
`META_BLOCKGUI`	`True`: The user interface is blocked while the script is being executed `False`: The user interface is not blocked while the script is being executed
`META_CYCLE`	Cycle (1 to 8) in which the script is to run

You can find additional information in section: Composing scripts (Page 338).

Note

When you archive a project, only the scripts located in the simulation project are archived.

External scripts are not archived with the project. The archived project is therefore incomplete.

6.3 Script syntax

6.3.1 Controlling the script

General

You can exit a running script using a instruction in the script or by initiating a dialog during the script runtime to display various processing options.

Canceling the script

The instruction
`BREAK`

is used to cancel the processing of a script. If the *BREAK* instruction is used in an included script, processing resumes in the calling script.

Terminating the script

The instruction
`QUIT`

is used to terminate the script. If the *QUIT* instruction is in an included script, the calling script will also be terminated.

The query dialog

The instruction
`DIALOG "text" MODE`

is used to pause the simulation and a query dialog with the content text is opened. The *Mode* parameter can be used in conjunction with the "Buttons for dialogs" table to determine which buttons this dialog will have.

Table 6-2 Buttons for dialogs

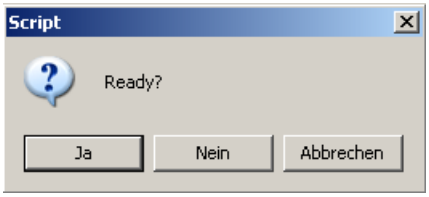
Mode	Button			
	"Yes"	"No"	"OK"	"Cancel"
YESNOCANCEL	X	X	–	X
OKCANCEL	–	–	X	X
YESNO	X	X	–	–
OK	–	–	X	–

YESNOCANCEL is the default mode. Selecting "Cancel" in the dialog box terminates the script; the simulation stays paused. In all other cases, the simulation is restarted and script execution resumes.

In the example

DIALOG "Ready?" YESNOCANCEL

the dialog box shown in the following figure appears.



If a log file was already open in the script before the *DIALOG* instruction, the system records in this file whether the dialog was closed with the "Yes" or "No" button. Based on this, the entry
DIALOG text True

or
DIALOG text False

is made in the log file.

In the script, a *YESNOCANCEL* dialog with the binary variables
`_result`

and
`_ok`.

can be used to query which decision the user has made.

The variable `_result` has the value *True* if the most recently opened dialog was closed with "Yes", and *False* if it was closed with "No". The variable `_ok` is also set to *False* if a dialog was acknowledged with "No". This variable is not automatically reset to *True*, but this can be performed using an instruction if desired.

6.3.2 Composing scripts

You can compose a script from several separate scripts. The instruction below means that the current script will continue with another script:
INCLUDE "Name"

The name of the script is entered relative to the calling script. Backslashes in the name must be doubled.

Example:
INCLUDE "..\\subscripts\\test2"

The *INCLUDE* instruction is executed once the included script has ended.

Multiple scripts can also be linked into chains in which the included scripts also contain *INCLUDE* instructions. The maximum depth of such a chain is limited to 10.

External scripts can also be included by specifying an absolute path and the name of the script file:
INCLUDE "C:\\scripts\\test3"

The included script file must have the file extension *.script*. The file extension is not specified in the file name of the *INCLUDE* instruction.

Note

Once a script is started, an executable instruction list is created for this script and all the included scripts. The included scripts must therefore exist before the calling script is started; changes made to an included script after the calling script is started have no effect.

Note

When you archive a project, only the scripts located in the simulation project are archived.

External scripts are not archived with the project. The archived project is therefore incomplete.

6.3.3 Commenting scripts

Comment lines start with two forward slashes. They are ignored when the script is executed.

Example:

// This is a single line comment

You can also comment out entire areas in the script by starting a comment with **/*** and ending it with ***/**.

Example:

/* This is a

multiline comment */

You cannot close one comment area and open a new one in the same line.

6.3.4 Signals in scripts

You can use scripts to access inputs, outputs, states and simulation model parameters that can be changed online. Signals are identified in SIMIT with the *source* of the signal and the *signal name*. The signal designation comprises both items separated by a forward slash and framed by quotation marks as follows:
`"Source/signal name"`

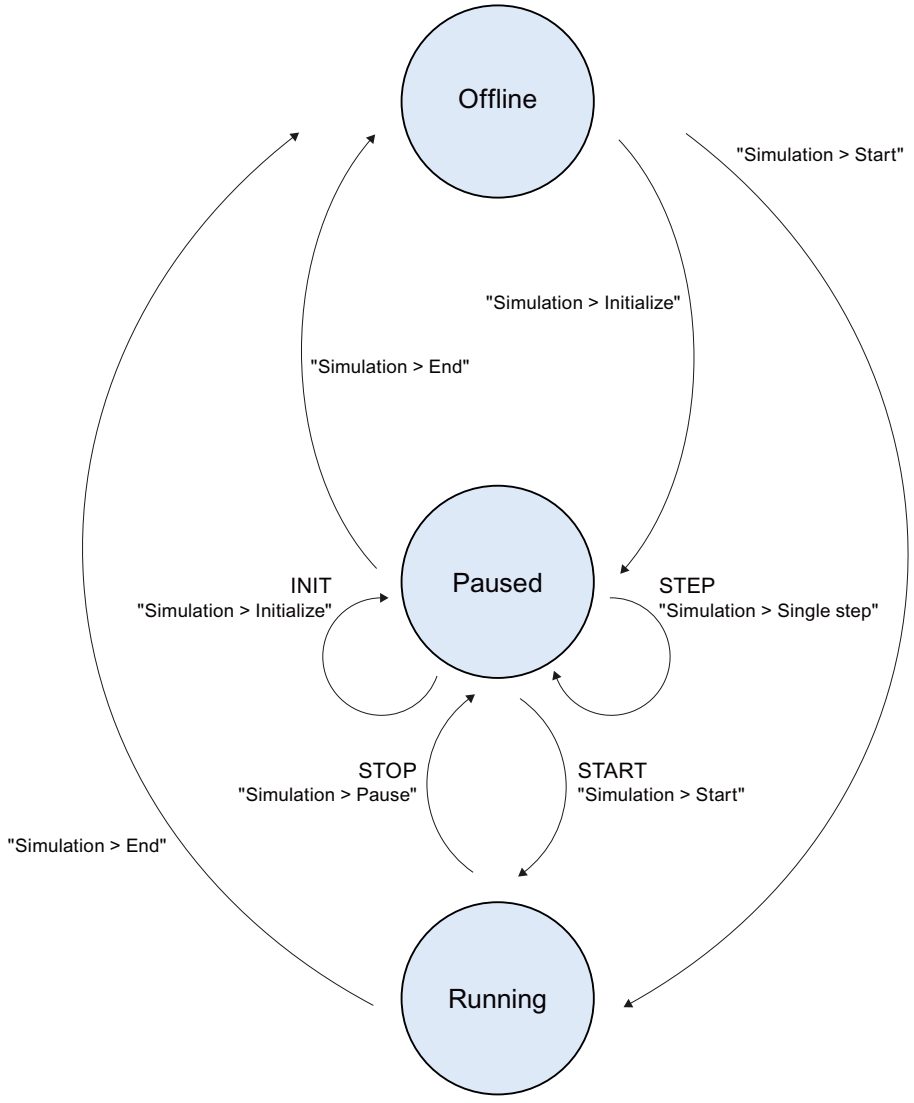
Example:
`"ADD#1/IN1"`

The following rules apply in the unusual event that the source or signal name contains double quotes or forward slashes:

- All double quotes must be preceded by a '\'.
- If the signal name itself contains a forward slash, then all forward slashes except for the delimiter between source and signal must be preceded by a '\'.
- If the source name contains a forward slash but not the signal name, then no marking is required. This is due to the fact that the last forward slash in the signal name is assumed to be the delimiter which would result in a correct result in this case.

6.3.5 Controlling the simulation

A simulation is always in one of the following three states: "Offline", "Stopped" or "Running".



The simulation state can be changed by a script. For example, the simulation can be stopped and started again. However, script commands are only effective if the simulation state is "Stopped" or "Running". Scripts cannot start an "Offline" simulation, nor can they end a simulation.

6.3.5.1 Initializing a simulation

The instruction
`INIT`

is used to initialize the simulation. This instruction corresponds to the command "Simulation > Initialize" in the SIMIT menu.

This command is invalid if the simulation has already been started. In this case, the following error message appears in the status bar and the script is terminated:

Wrong simulation state, Script '...', Row ... .

Note

The *INIT* script command cannot cause the transition from "Offline" to "Online", because a script can only be executed in online mode.

6.3.5.2 Starting the simulation

The instruction
`START`

is used to start the simulation. This instruction corresponds to the command "Simulation > Start" in the menu or the command with the button in the SIMIT toolbar.

A *START* instruction in the script is ignored if the simulation has already been started.

Note

The *START* script command cannot cause the transition from "Offline" to "Online", because a script can only be executed in online mode.

Starting with this command automatically sets the simulation time to real-time (100%).

6.3.5.3 Starting and waiting until an absolute time

The instruction
`GO-TO time`

is used to pause the script until the simulation time is equal to or exceeds *time* in milliseconds.

The instruction is ignored if the simulation time is already equal to or greater than *time*. If the simulation is initialized but has not yet been started, this command will cause the simulation to begin.

6.3.5.4 Starting and waiting until a relative time

The instruction
`GO-FOR time`

is used to pause the script until *time* in milliseconds has passed.

If the simulation is initialized but has not yet been started, this command will cause the simulation to begin.

6.3.5.5 Starting and waiting for a certain number of time slices

The instruction
`GO n`

is used to pause the execution of the script for *n* time slices. The time slices are counted in the script time slice. For one cycle, i.e. for `n=1`, the parameter *n* can be left out.

If the simulation is initialized but has not yet been started, this command will cause the simulation to begin.

6.3.5.6 Starting and waiting for an event

The instruction
`GO-UNTIL Condition TIMEOUT time`

is used to pause the script until the *Condition* is met or the *TIMEOUT* time specified in the simulation in milliseconds has expired.

Example:
`GO-UNTIL "OR#1/Y" TIMEOUT 10000`

You do not have to enter a time value for *TIMEOUT*. The script would then no longer close of its own accord if the condition was not met.

If the simulation is initialized but has not yet been started, this command will cause the simulation to begin.

6.3.5.7 Stopping the simulation

The instruction
`STOP`

is used to stop the simulation. This instruction corresponds to the command "Simulation > Pause" in the menu.

A *STOP* instruction in the script is ignored if the simulation has already been stopped.

6.3.5.8 Executing a single step

The instruction
`STEP`

is used to carry out a single step in the simulation. This instruction is the same as the "Simulation > Single step" command in the SIMIT menu. The function key F12 is assigned to this function.

This instruction is invalid if the simulation is already running cyclically. Otherwise the following error message appears in the status bar and the script is terminated:

Wrong simulation state, Script '...', Row ... .

Note

A simulation step is always one step of the fastest submodel, which means the submodel with the shortest cycle time.

6.3.5.9 Saving a snapshot

The instruction
`SAVE-IC "name"`

is used to save a snapshot in the simulation project under the name *name*.

The snapshot is always saved at the highest level of the "Snapshot" folder. The specifying of subfolders is not permitted.

NOTICE
Data loss Any snapshot with the same name that already exists in your simulation project will be overwritten without a prompt and the data saved in it will be lost.

6.3.5.10 Loading a snapshot

The instruction
`LOAD-IC "name"`

loads the snapshot with *name* from the "Snapshot" folder of the simulation project.

The specifying of subfolders is permitted. Any backslashes in the name must be doubled.

Example:
```
LOAD-IC "name"
LOAD-IC "subfolder\\name"
```

If no snapshot with this name exists, the following error message appears in the status bar and the script is terminated:

Snapshot '...' cannot be loaded, the file might not exist, Script '...', Line ... .

After a snapshot is loaded, the simulation is stopped and must be restarted using either the *STEP* or *START* instruction.

6.3.5.11 Resetting the simulation time

The instruction
`SIMTIME-RESET`

is used to reset the simulation time to zero. The simulation state remains unchanged.

Note

Resetting of the simulation time can lead to undesired results if simulation mechanisms are directly dependent on the simulation time, e.g. the trend display, messaging editors or isochronous operation.

6.3.6 Logging

You can log the results of your simulation, i.e. signal values or events. Script instructions can be used to create log files in which outputs are documented.

Script instructions to output data are ignored if no log file is open when the script is executed.

6.3.6.1 Opening and closing log files

The instruction
`OPEN-LOG "name"`

is used to open a log file with the name *name*. Any log file that may already be opened is closed first. The log file must be entered with its absolute path. Any backslashes in the name must be doubled.

Example:
`OPEN-LOG "c:\\protocol\\log.txt"`

Note

Any log file that already exists with the same name will be overwritten without warning. Data that is already saved in the log file will be lost.

You must close the log file either at the end of the script or at the point where you no longer wish to record any more data.

The instruction
`CLOSE-LOG`

closes the log file.

Note

The outputs are not saved in the log file if the log file was not closed by a instruction before the end of a script. In this case, the log file is empty.

6.3.6.2 Unformatted output

The instruction
`PRINT "Signal"`

saves the current value of the signal *Signal* in the open log file. Inputs, outputs, states and component parameters that can be changed online may be used as signals. Binary values are output as True or False.

6.3.6.3 Formatted output

The instruction
`PRINTF "Formatstring", "Signal1", "Signal2", …`

saves the current values of the signals listed in the instruction in the open log file. The *Formatstring* contains the format specifier for each signal; it must contain precisely one format specifier for each signal. The format string may also contain text. The allowed format specifiers are listed in the table below.

Table 6-3 Formatting instructions

Signal type	Formatting instruction
analog, integer, binary	%f or %.nf (n: number of decimal places), binary values are output as "0" or "1".
integer	%i
binary	%b (values are output as "True“ or "False“)

A backslash must be placed in front of the percentage sign in the format string if you would like the percentage sign to be output as a normal character: **\%**. Use the *PRINTF* instruction without a signal specification if you want to output a fixed text only.

Signals can be any input signals, output signals, states or component parameters that can be changed online. You can find additional information on this in the section: Signals in scripts (Page 339).

In the example
`PRINTF "Ramp: %.2f [ULR=%b] [LLR=%b]", "Ramp#1/Y", "Ramp#1/ULR", "Ramp#1/LLR"`

the following is output to the log file:
`Ramp: 0.00 [ULR=False] [LLR=True]`

6.3.6.4 Outputting time and date

You can enter the time and date of your computer into the log file using the system variables listed in the following table:

Table 6-4 System variables

Syntax	Meaning
_t_day	Day
_t_mon	Month
_t_year	Year
_t_hour	Hour
_t_min	Minute

Syntax	Meaning
_t_sec	Second
_t_msec	Millisecond

All figures appear as two digits.

For example, the following instructions

```
PRINTF "Date:   %.0f.%.0f.%.0f", "_t_day", "_t_mon", "_t_year"
PRINTF "Time:  %.0f:%.0f:%.0f", "_t_hour", "_t_min", "_t_sec"
```

generate the following entries in the log file:

Date: 20.06.11

Time: 12:55:13

Note

The system variables in the "System variables" table can only be used in the context of logging and cannot be used for general calculations within the script.

Note

If the system time or system date is changed, the system variables `_t_hour`, `_t_min`, `_t_sec` and `_t_msec` are not updated during the active simulation. If the simulation is restarted, these system variables are updated again.

6.3.6.5 Outputting version information

You can output the version of your current project and the currently used SIMIT version to the log file using the system variables *_ProjectVersion* and *_SIMITVersion*.

For example, the following instructions

PRINTF "Project version: %s", "_ProjectVersion"

PRINTF "SIMIT version: %s", "_SIMITVersion"

generate the following entries in the log file:

Project version: AA12345-344332-3.4

SIMIT version: 7.1.0

6.3.6.6 The _printlog system function

It is also possible to write directly from components to an open log file. However, to do this you will have modify the behavior description of the relevant component types, which means the system function.

```
_printlog("string");
```

must be inserted in order to output text. When called, this function writes the text in the *string* directly to the open log file.

Note

The SIMIT component type editor (CTE) is required to edit the behavior description of component types. The component type editor is a SIMIT expansion module with which you can create your own component types.

6.3.7 Signal curves

6.3.7.1 Opening and closing a plot file

You can define signals in the script whose values are to be written cyclically to a plot file. Only one plot file can be open in a script at any one time.

The instruction
```
OPEN-PLOT "File name"
```
opens the plot file *File name* and starts to record the signals. The name of the file (*File name*) must be specified with its absolute path, in which any backslashes must be doubled.

A plot file that is already open is closed first. If the file *File name* already exists, it will be overwritten without confirmation.

Note

The signals to be recorded must be specified before the file is opened.

The first line of the plot file shows the signal names of the signals to be recorded. The remaining rows contain the simulation time in milliseconds, and tab-delimited signal values in the sequence that is specified in the first row.

The instruction
```
CLOSE-PLOT
```
closes the plot file and the list of signals to be recorded is deleted.

The instruction sequence
```
STOP
INIT
START
PLOT "Ramp#2/Y"
PLOT "Ramp#2/ULR"
PLOT-CYCLE 500
OPEN-PLOT "D:\\plot.txt"
GO-FOR 4000
CLOSE-PLOT
```

would add the following to the plot file:

	Ramp#2/Y	Ramp#2/ULR
0	1.66666666666667	Ramp#2/ULR
500	18.3333333333333	False
1000	35.0	False
1500	51.6666666666667	False
2000	68.3333333333333	False
2500	85.0	False
3000	100.0	True
3500	100.0	True
4000	100.0	True

6.3.7.2 Specifying signals

The instruction
`PLOT "Signalname"`

adds the signal *Signalname* to the list of signals to be recorded. Please note that the list of signals to be output must be completely created before the plot file is opened.

You can record input signals, output signals, states and component parameters that can be changed online within your simulation project.

6.3.7.3 Specifying a cycle

The instruction
`PLOT-CYCLE time`

specifies the cycle in which the signals are to be recorded. The cycle time *time* specified in milliseconds is rounded up to the next multiple of the script cycle time. If you do not add this instruction to your script, the script cycle time is used for the recording. Please note that the cycle must be specified before the plot file is opened.

6.3.8 Setting signals

You can set input signals, status variables and parameters and components that can be changed online with instructions in the script. You can also set these variables to pre-defined values or apply a linear change to the values.

Note

If you set inputs that are linked to the outputs in the simulation project or give values for states of a component that are set in the component itself, the value specified in the script will either be completely ignored or will only be effective during one computing time slice.

6.3.8.1 Setting individual values

The instruction
`"Signal" = Value`

sets the specified *Signal* to the given *Value*.

Example:
`"Status/1/BI" = True`

Instead of a constant value, you can also specify an expression for the value. An expression consists of constants or signals that are linked to each other by operators. The permitted arithmetic operators are listed in the "Arithmetic operators" table, and the permitted Boolean operators are listed in the "Boolean operators" table.

Table 6-5 Arithmetic operators

Operator	Meaning
( )	Parentheses
* /	Multiplication Division
+ –	Addition Subtraction

Table 6-6 Boolean operators

Operator	Meaning
( )	Parentheses
!	not
&&	and
||	or

The operators are listed in both tables from highest to lowest priority.

Example:
`"Display/E" = 2 * ("Slider/A" + 0.5)`

Note

A value that you have assigned to a signal will not be available until the next execution time slice. If you have set a signal using an instruction in the script, for example, and you want to use this signal to assign a value or output its value in the instruction that immediately follows, the signal still has the value that it had at the beginning of the cycle.

6.3.8.2 Separating connected signals

You can use the
`FIX "Signal"`
instruction to disconnect the specified signal ("force") to then be able to set a value.

Example:
`FIX "Status/1/BI"`

```
"Status/1/BI"=True
```

You can use the
`UNFIX "Signal"`
instruction to revoke the disconnection of the specified signal.

Example:
`UNFIX "Status/1/BI"`

The two commands, FIX and UNFIX, correspond to the operation of the signal splitter on the user interface.

6.3.8.3 Specifying signals

The instructions
`RAMP "Signal" FROM start TO end IN time`
`RAMP "variable" TO end IN time`

allows you to set an *analog* input signal or the state of a component to a linearly rising or falling value. This instruction sets the value *start* and causes the value to increase or decrease so that it reaches the value *end* within *time* milliseconds it, which means the value changes linearly (ramp).

There is no start value (FROM start) in the second instruction. The start value in this case is the current value.

Script execution continues during the specified time *time*, which means other instructions in the script are executed without waiting for the signal to reach its specified end value. A script is not terminated until all signals that were set using ramps have reached their end values.

Examples:
`RAMP "Display#1/X" FROM 20.0 TO 10.0 IN 15000`

6.3.9 Conditional execution

Basics

You can implement conditions for execution with the following instructions:

```
IF expression THEN
    block1
ELSE
    block2
ENDIF
```

If the condition *expression* is true, the instruction list *block1* is executed; otherwise list *block2* is executed. The two instruction lists consist of one or more lines with any number of instructions, and can therefore themselves contain conditional instructions.

The *ELSE* instruction can also be omitted:

```
IF expression THEN
    block
ENDIF
```

You must close each IF instruction with ENDIF.

The *expression* condition allows you to access input signals, output signals, states and component parameters that can be changed online. The following relational operators are supported:

Table 6-7 Comparison operators

Operator	Meaning
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
!=	Not equal to

A condition can consist of multiple expressions that are linked with Boolean operators. You can find additional information on Boolean operators in the tables in: Setting individual values (Page 349).

Examples

```
IF "Button/1/Z" == True THEN
    "Display/E" = 2.0 * ("Slider/A" + 0.5)
ELSE
    "Display/E" = "Slider/A"
ENDIF
```

```
IF "Button#1/Z" THEN
    "Display/E" = 2.0 * ("Slider/A" + 0.5)
ELSE
    IF !"Button#2/Z" THEN
        "Display/E" = 3.0 * ("Slider/A" + 0.5)
    ELSE
        "Display/E" = "Slider/A"
    ENDIF
ENDIF
```

Comparing analog signals

The values of analog signals are mapped in floating point format. Rounding can produce different values which then lead to incorrect comparison results. Use the `RANGECHECK` function for the value comparison with analog signals. The syntax of the function is as follows:

- RANGECHECK("Analog value", "Setpoint", "Tolerance" | "Lower tolerance", "Upper tolerance")
  - Analog value: Value of the analog signal that is compared with the "setpoint"
  - Tolerance: Amount by which the analog value is permitted to deviate from the "setpoint". You can alternatively specify different tolerance values for the low and high limits.

The function returns True or False.

In the example below, "True" is returned when the value of the analog signal "TemperatureValue" is between "0.5" and "1".

```
IF RANGECHECK("TemperatureValue", 0.75, 0.25) == True THEN
    Block1
ELSE
    Block2
ENDIF
```

In the example below, "True" is returned if the value of the analog signal "TemperatureValue" is between "0.5" and "1.25".

```
IF RANGECHECK("TemperatureValue", 0.75, 0.25, 0.5) == True THEN
    Block1
ELSE
    Block2
ENDIF
```

6.3.10 Accessing the simulation time

The system variable
`_Time`

contains the current simulation time in milliseconds.

In the example
```
IF ("_Time" > 20000) THEN
    PRINTF "Simulation time: %.0fms", "_Time"
ENDIF
```

the output sent to the log file might be:
```
Simulation time: 26600 ms
```

Component type editor

7

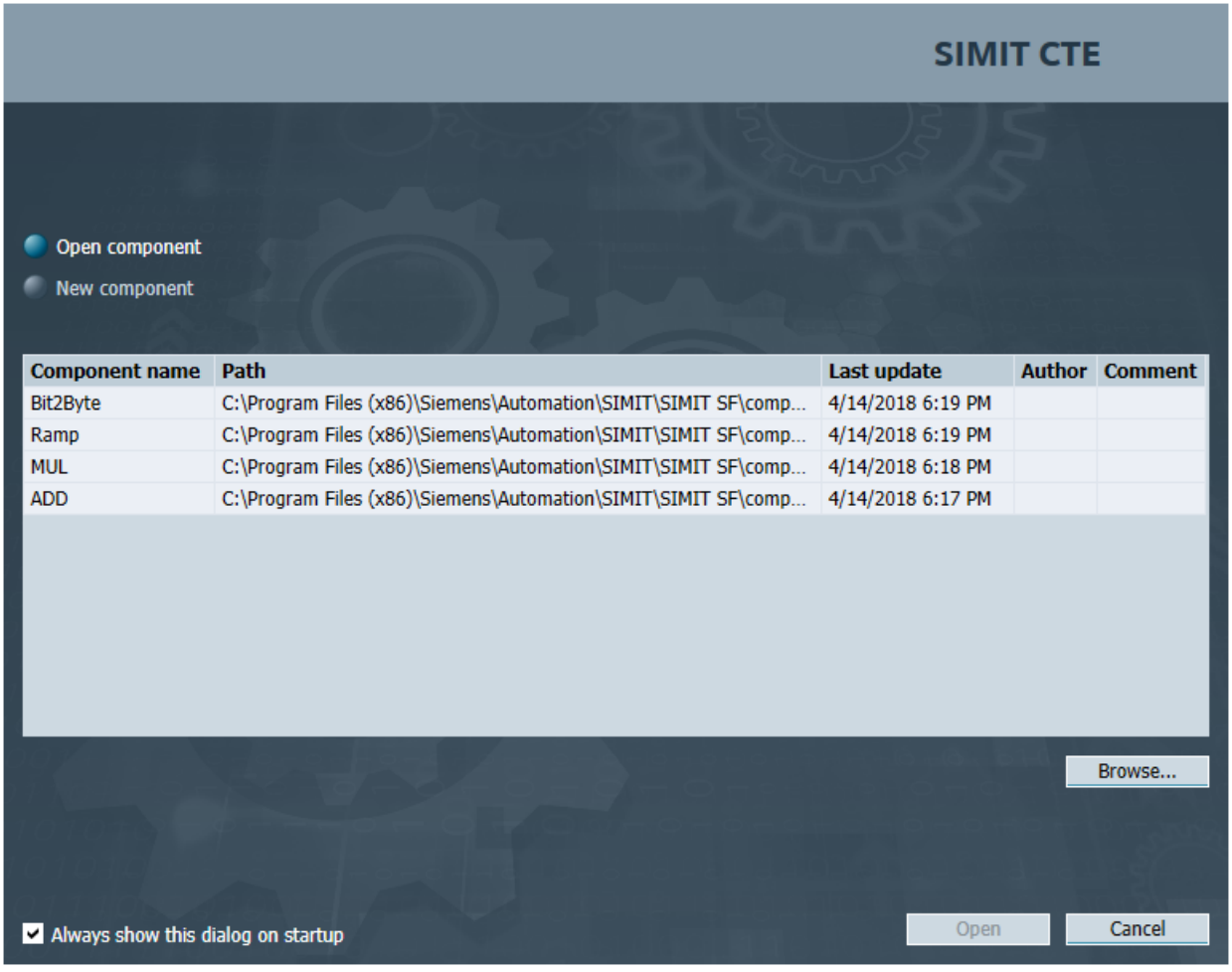
7.1 User interface

7.1.1 Starting the CTE

The Component Type Editor (CTE) is a stand-alone SIMIT application. You start the editor using the Start menu in the folder **Programs > SIMIT > SIMIT CTE**. Once it has started, you have the choice of opening an existing component type or creating a new component type. You can also access this dialog at any time from the **Component** menu.

Note

In the CTE user interface, the term 'component' is used for short instead of 'component type'.



Starting the CTE from SIMIT or using a component file

You can also open component types for editing from the SIMIT *Components* task card. To do this, select the required component type and double-click it. You can also use the *Open* command in the shortcut menu of the component type you wish to open. If the CTE was not yet running, it will automatically start to open the component type.

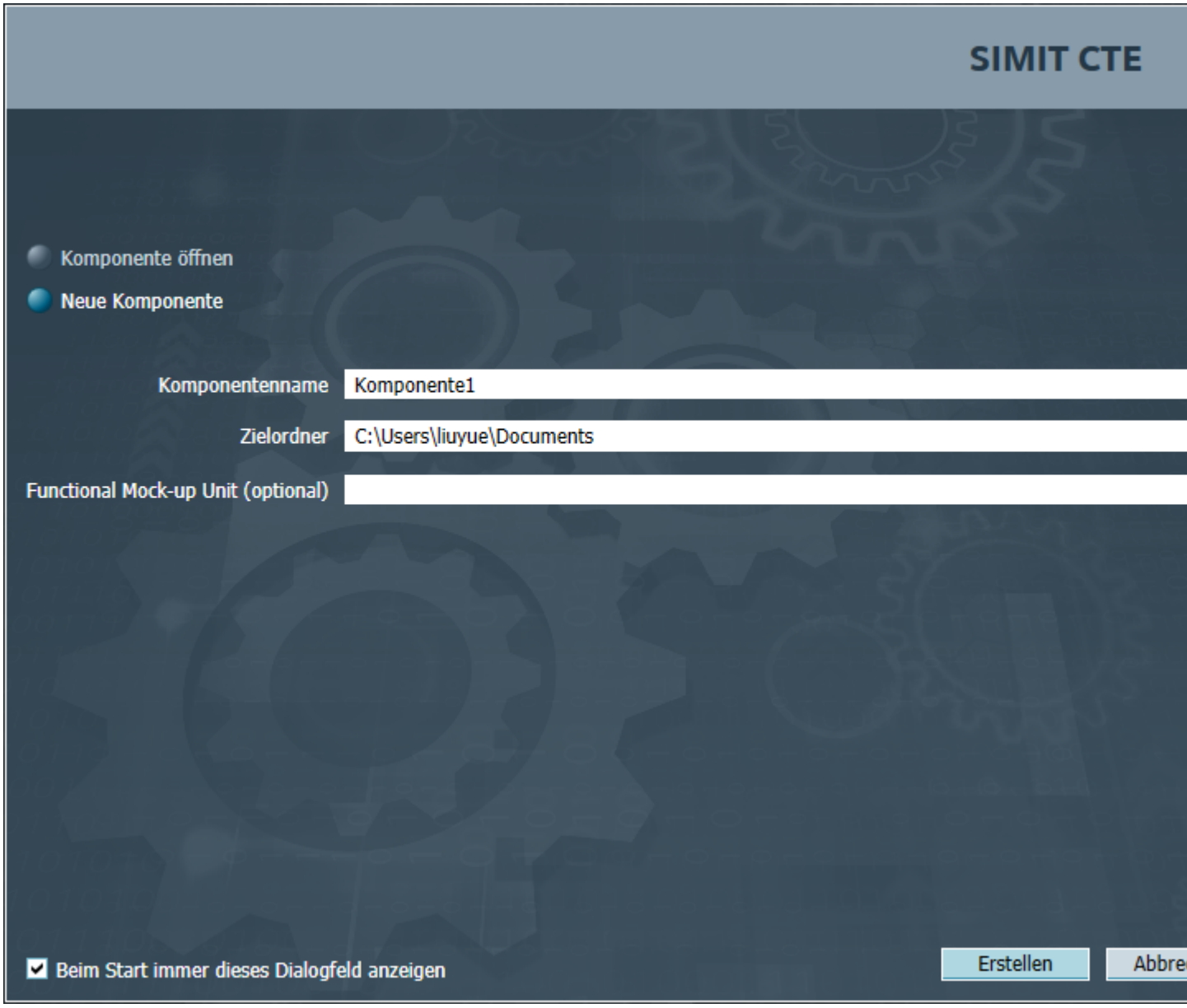
Component types are stored on the file system in a file with a name ending in *simcmp*. You may also open a component type in the CTE by double-clicking the file. Here, too, the CTE is launched to open the component type if it was not running already.

Creating a new component

Once the CTE is started, you can create a new component.

SIMIT CTE provides the option of importing a Functional Mock-up Unit (FMU) and converting it to a SIMIT component type.

Functional Mock-up Units (FMU) (Page 412)



7.1.2 Structure of the user interface

The CTE user interface is based on the SIMIT GUI design. It is subdivided into the following panes:

The **menu bar** and **toolbar** allow easy access to the CTE functions. There are additional functions available in the shortcut menus.

The **project windoww** shows the open component type in a tree view.

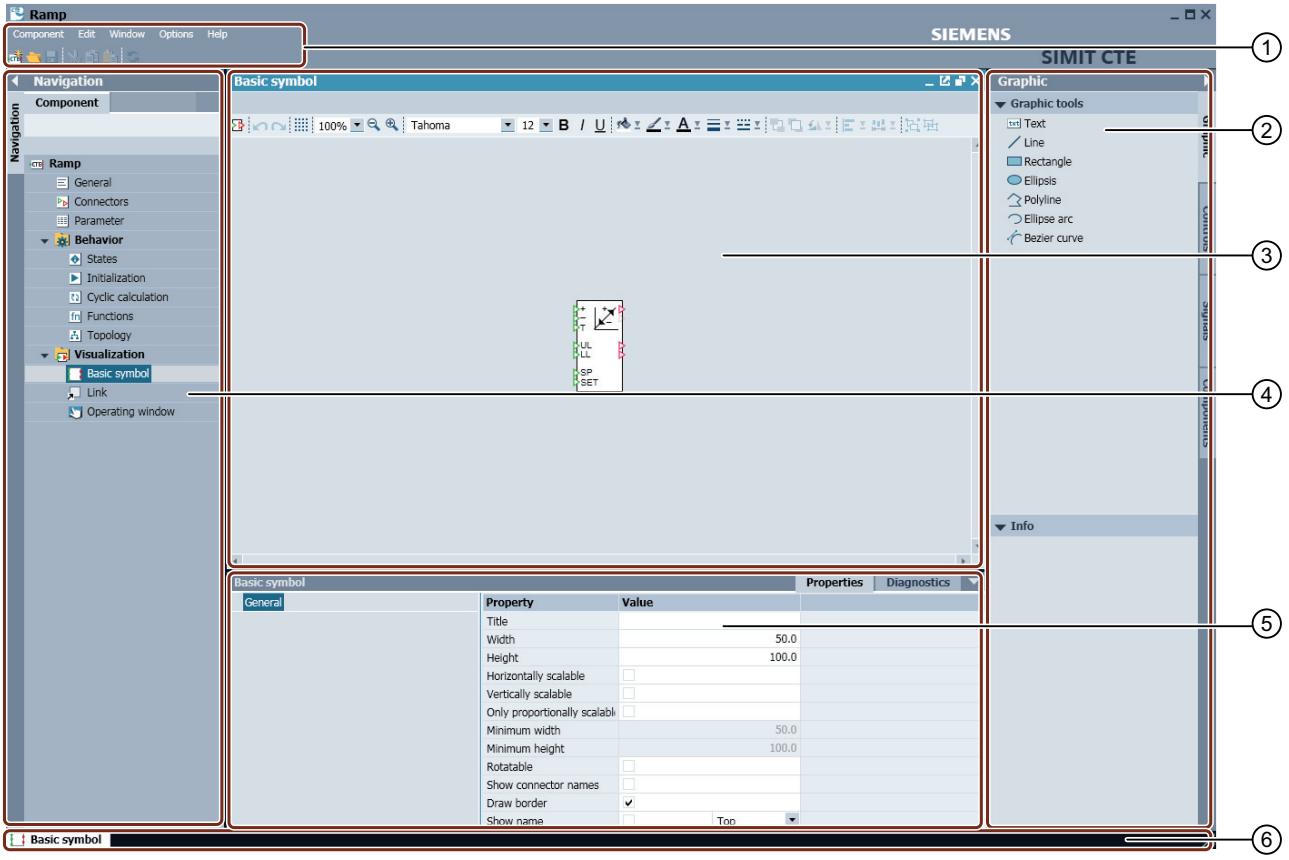
The editors are opened for editing in the **work area**. Every editor contains a toolbar for quick access to the editor-specific functions.

The **Tool window** contains the tools that can be used with the respective editor, such as connection types and graphic tools in task cards.

The **Properties window** shows the properties of an object selected in the work area.

The **editor bar** at the bottom left of the interface allows you to toggle between opened editors.

The **status bar** at the bottom right of the interface shows information about the current status of the CTE.



① Menu bar and toolbar
② Tools window
③ Work area
④ Project window
⑤ Property view
⑥ Editor bar, status bar

All editors are opened in the work area. The tools window only contains the specific task cards for each editor. There are menu commands that divide the work area horizontally (*Window > Split horizontally*) or vertically (*Window >Split vertically*) so that two editors can be opened side by side or one below the other in the work area.

7.1.3 Menu bar and toolbar

The CTE menu bar contains all commands you need to create or open and edit component types.

Frequently used functions are also provided on the toolbar. These include the following functions:

 (New Component) for creating a new component type

 (Open ...) for opening a component type

(Save) for saving a component type

 (Cut) for cutting selected objects

 (Copy) for copying selected objects

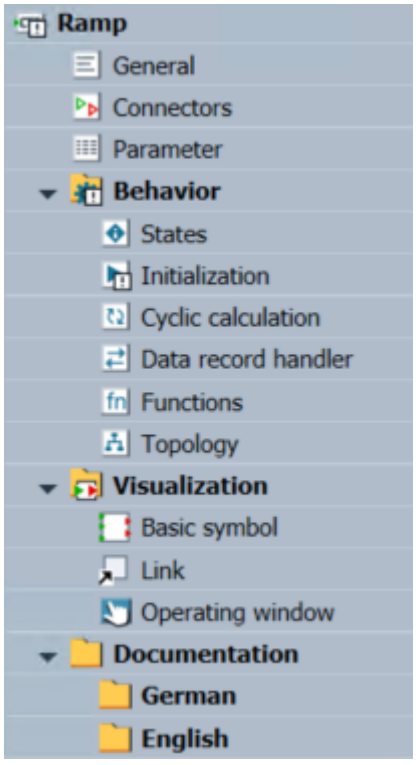
 (Paste) for pasting copied objects

 (Refresh) for updating a component type.

Some aspects of a component type affect one another. The *Refresh* function ensures that all information in one aspect is compared with the information in the other aspects. When you refresh, there is also a check to ensure that the specification of the component is formally correct.

7.1.4 Project tree

The project tree lists all aspects of a component type. For every aspect an associated editor can be opened by double-clicking the relevant entry in the project tree. Formal errors in the implementation of an aspect are identified by an overlay in the project tree: . All higher levels of this aspect are identified with this overlay as well.



7.1.5 Keyboard shortcuts

You can use the keyboard shortcuts listed in the table below to speed up the editing of a component type. All keyboard shortcuts are context-specific, which means they can only be used if the associated editor has the keyboard focus.

Table 7-1 Keyboard shortcuts

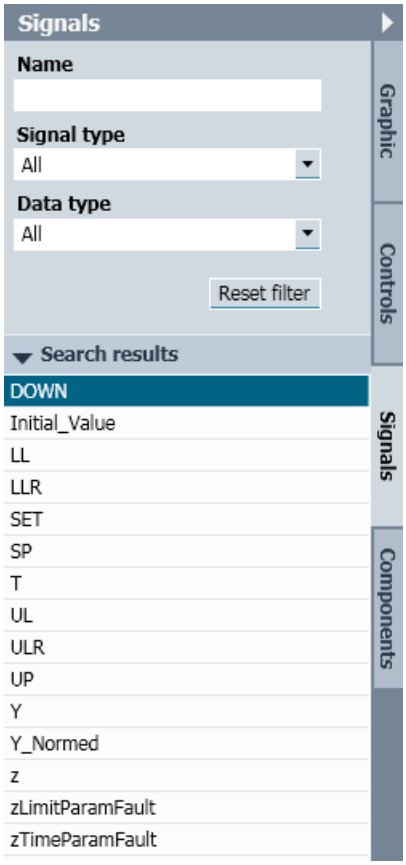
Shortcut keys	Meaning
Ctrl + A	Select all
Ctrl + C	Copy
Ctrl + F	Find
Ctrl + H	Replace
Ctrl + S	Save
Ctrl + V	Paste
Ctrl + X	Cut
F2	Rename

Shortcut keys	Meaning
F3	Find Next
F5	Refresh

7.1.6 The task cards

7.1.6.1 The "Signals" task card

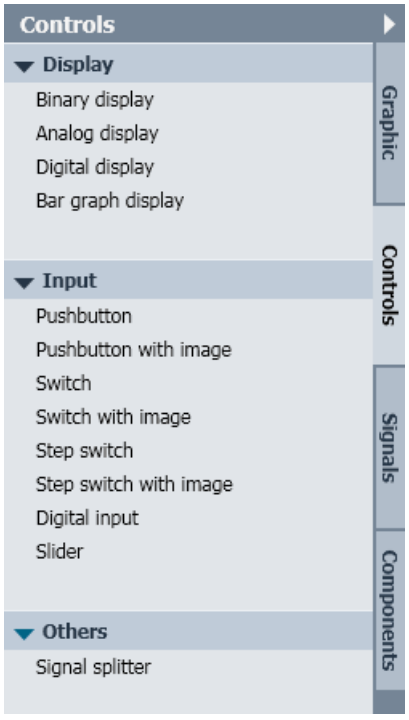
For every text editor of the behavior description, there is a "Signals" task card. It contains all signals that exist in this component type. Here the "source" of signals is the component type itself, so you only need the name for the signals in this case.



The *Signals* task card allows you to filter by name, signal type and data type. It thus provides a rapid overview of the signals available in this component type. You can also easily move a signal name from the task card into the text editor using drag & drop.

7.1.6.2 The "Controls" task card

The "Controls" task card provides all controls from the SIMIT basic library.



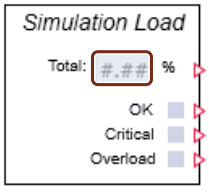
There are three panes of controls:

- Controls for displaying signal values in the *Display* pane
- Controls for entering signal values in the *Input* pane and
- Other controls in the *Others* pane

Controls can be found in the library under their names.

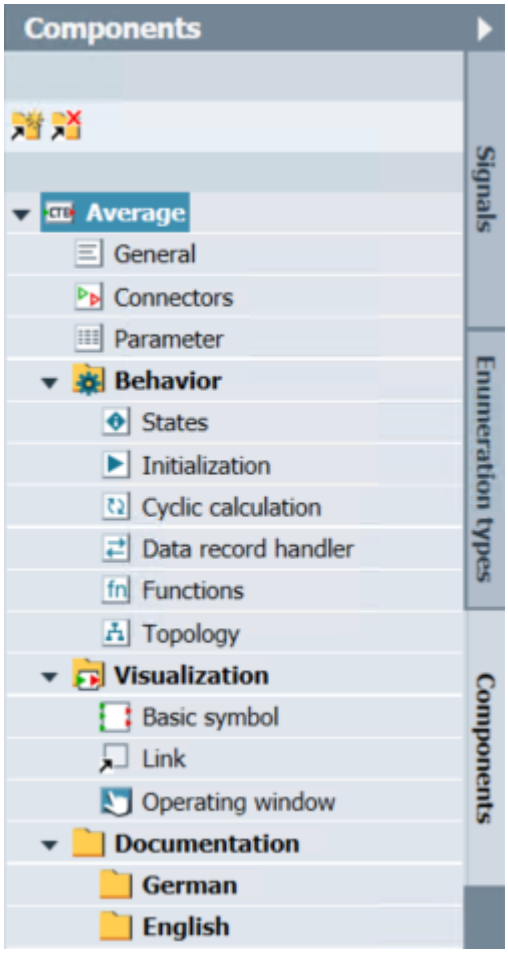
You may place controls on the basic symbol and connect them to input or output signals.

For component types that use controls on the basic symbol, you can set and display values directly on the symbol without having to open the operating window of the component type.



7.1.6.3 The "Components" task card

You can open additional components in read-only mode in the "Components" task card.



Click on the symbol to open a component and display its properties.

Click on the symbol to close the selected component.

You can open and copy individual objects of a component in the corresponding editor and insert them into objects of another component. You can open several components simultaneously and thus create a component consisting of parts from several other components.

Note

You can protect components you have created yourself with a password. You are prompted for the password when you open such a component. Some components from the basic library are protected when they leave the company and cannot be opened. You can find additional information in section: Protection of the component type (Page 369).

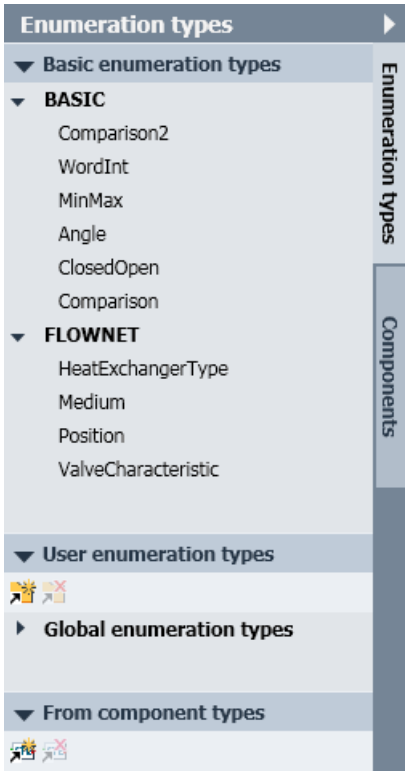
You can find additional information on the properties of a component in section: Properties of component types (Page 366).

Transferring a component to SIMIT

When you create custom component types with the CTE, you have to save them either in a library directory or under *Global Components* so that you can access them in SIMIT. SIMIT updates the pane of the *User Components* automatically when you save a component type with the Component Type Editor in it.

7.1.6.4 The "Enumeration Types" task card

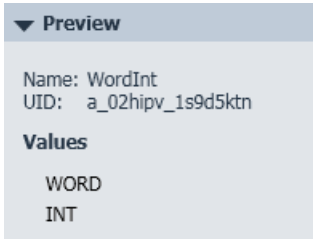
The "Enumeration types" task card contains all enumeration types that can be used for enumeration parameters.



This task card is divided into three panes:

- **Basic enumeration types**
  This pane shows the enumeration types that are used in the components of the basic library. You can use these enumeration types as the basis for your own enumeration types by copying them to the *User enumeration types* pane.
- **User enumeration types**
  You can create your own enumeration types in this pane. To do this, copy an existing enumeration type or click the *New enumeration type* command. A window then opens in which you define the enumeration type.
  Enter the names for the individual elements of the enumeration.
- **From Component Types**
  You can open any component type from this pane on the task card using the command. The enumeration types used are displayed under this component type and in the selection list.

When you select an enumeration type from one of these three panes on the task card, the elements that can be contained in this enumeration type are listed in the preview at the bottom of the task card.

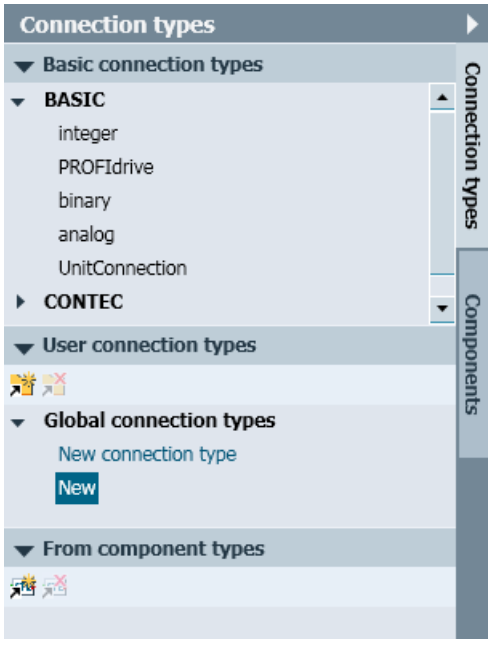


The unique identifier (*UID*) of the enumeration type is also displayed in the preview.

In the behavior description, you use an element of an enumeration type by entering the name of the enumeration type followed by a period and the name of the element itself. The entire construct must also appear in single quotes, for example: `'ClosedOpen.Closed'`.

7.1.6.5 The "Connection Types" task card

The "Connection types" task card lists all connection types known in SIMIT and allows you to define custom connection types.



This task card is divided into three panes:

Basic connection types

This pane shows the connection types that are used in the component types of the basic library. These are essentially the basic connection types as described in the table below.

Table 7-2 Basic connection types

Connection type	Value	Value range
binary	Binary values	True/False
analog	Floating point values	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$
integer	Integer values	–9.223.372.036.854.775.808 bis +9.223.372.036.854.775.807

You can use the basic connection types as the basis for your own connection types. To do this, copy the connection type to the *User connection types* pane and edit it.

User connection types

This pane allows you to create your own connection types. To do this, copy an existing connection type or click the *New connection type* entry. A window then opens in which you define the signals.

You can create any number of signals in the *Forward* and *Backward* directions. For the signal type, you can only choose between the *analog*, *binary* and *integer* data types.

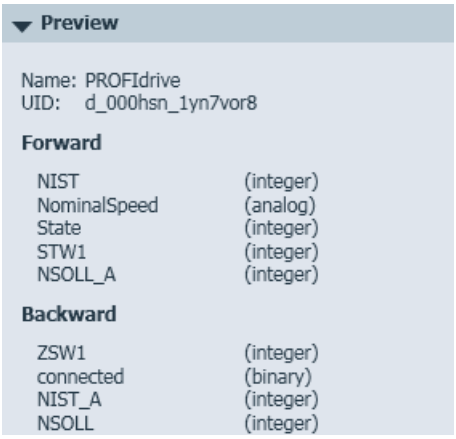
Select the "Multiple Connection" check box if a component output is to be interconnected with multiple inputs. Multiple connections are not permitted for connection types that contain backward signals, otherwise multiple output signals would be routed to the same input. You can thus only check the Multiple Connection check box if no signals are defined in the backward direction.

When you close the dialog, a unique identification number (ID) is assigned to this connection type.

From Component Types

When you create components and you want their connectors to be compatible with the connectors of other components, it is important to use the same connection type for them. To do this, you can open any component type from this pane on the task card using the  command. The connection types used are then displayed under this component type and in the selection list for connection types.

When you select a connection type from one of these three panes on the task card, the signals that can be transferred using this connection type are listed in the preview at the bottom of the task card:



The unique identifier (*UID*) of the connection type is also displayed in the preview. Note that connection types are only identical if they have the same UID. The name of the connection type is not a sufficient criterion. Even connection types for which the same signals are defined do not have to be identical.

7.2 Principles of component types

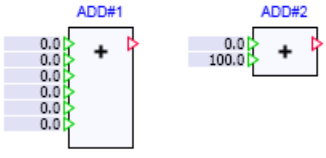
7.2.1 The SIMIT type-instance concept

Introduction

In SIMIT, components are the smallest units from which a simulation is assembled. All components are instances of types that are made available in libraries. Component types are created and edited using the Component Type Editor (CTE). This takes into account all aspects that can be used for components in SIMIT .

The type/instance concept

In SIMIT, the functional simulation model is made up of the functional behavior of the individual components that are positioned graphically on charts, assigned parameters and interconnected. SIMIT follows a type/instance concept in the process: The parameterizable function is defined in the type, while the individually parameterizable instances of the type are added to charts. We therefore speak of both component types and components as instances.



This type-instance concept allows you to change a component type without having to change the instances already created from it.

Note

After making changes to a component type, if you want to update the instances you have already created in your simulation project, you simply use the *Find&Replace* function in SIMIT to replace component types.

Every component instance is identified by a separate, unique name in SIMIT. Every instance can be parameterized individually with respect to both the actual parameters, any preassigned inputs and the symbol scaling.

7.2.2 Properties of component types

A component type is a discrete unit that can be created and modified with the Component Type Editor. From a technical viewpoint, a component type is a file with the extension *.simcmp*. SIMIT libraries are thus simply directories in your file system in which component types are stored for use.

All properties that can be used in SIMIT are implemented in a component type. The implementation of a component type comprises the following aspects:

- **General information**
  General information relates to the administration, protection and special features of a component.
- **Connectors**
  Connectors are all the visible and invisible signal inputs and outputs of a component type. The connectors are defined with their properties in the component type.
- **Parameters**
  Parameters are used to customize the individual component instances. The component type defines which of its properties should be parameterizable in the instance.
- **Behavior**
  The definition of status variables and the functional behavior description determine the functional behavior of a component. The dependencies between the output signals and the input signals and parameters are thereby described in detail.
- **Visualization**
  Components are graphically represented on charts with a basic symbol. Optionally, components may also have a symbol for a link and an operating window.

Every functional component type, which means one that can be used in SIMIT, is automatically assigned a unique identifier (ID) when it is saved with the CTE.

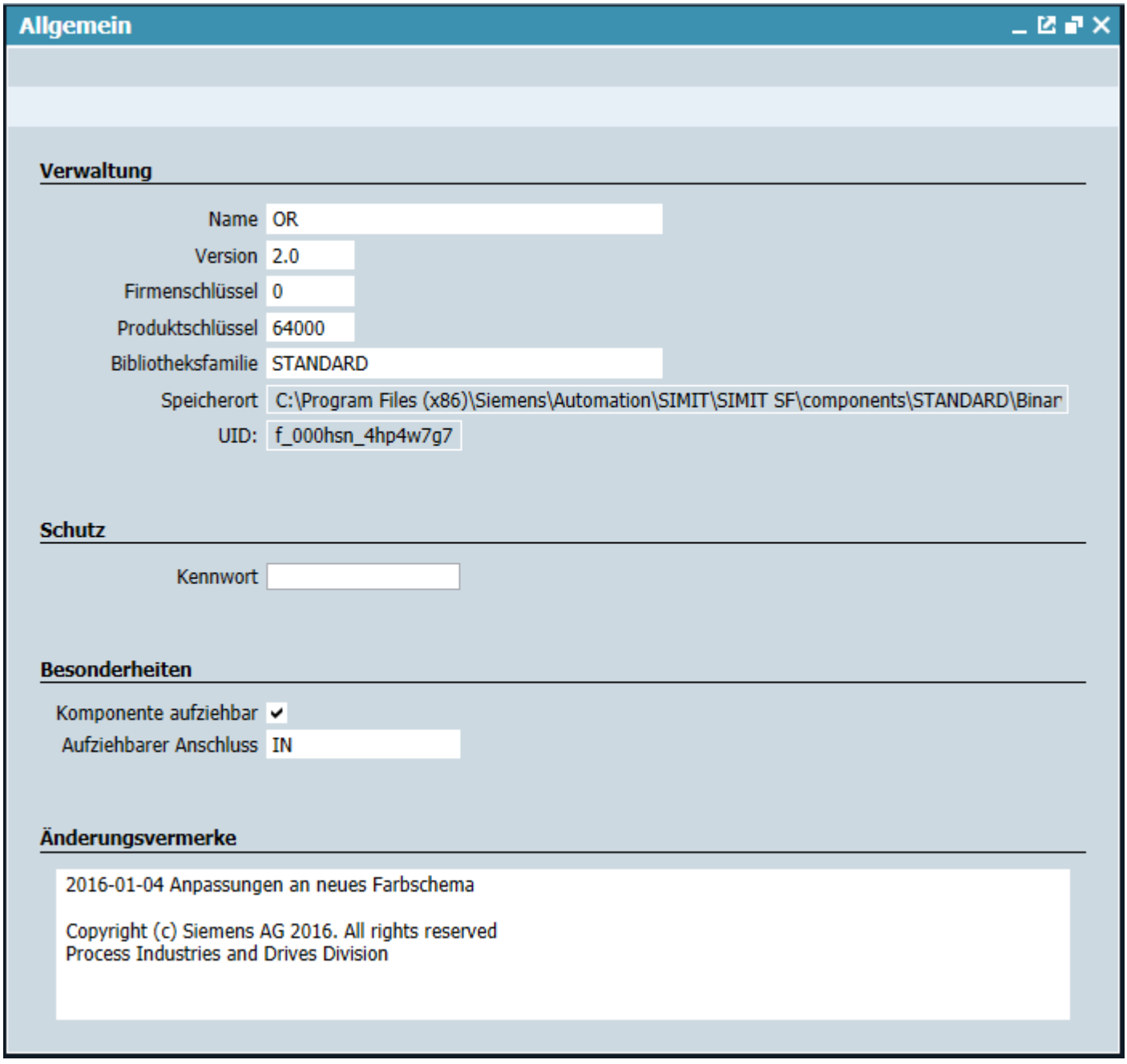
7.2.3 General properties of component types

7.2.3.1 Overview

General properties of a component type include

- Administration
- Protection
- Special features and
- Changes

of that component type. To edit the general properties open the corresponding editor by double-clicking the aspect *General* in the project tree. You can then edit the individual properties, which are presented in a fixed arrangement of groups of text boxes.


Allgemein
Verwaltung
Name OR
Version 2.0
Firmenschlüssel 0
Produktschlüssel 64000
Bibliotheksfamilie STANDARD
Speicherort C:\Program Files (x86)\Siemens\Automation\SIMIT\SIMIT SF\components\STANDARD\Binar
UID: f_000hsn_4hp4w7g7
Schutz
Kennwort
Besonderheiten
Komponente aufziehbar
Aufziehbarer Anschluss IN
Änderungsvermerke
2016-01-04 Anpassungen an neues Farbschema
Copyright (c) Siemens AG 2016. All rights reserved
Process Industries and Drives Division

7.2.3.2 Administration properties

Administration information is as follows:

- Name
  The "Name" of a component type can selected as desired. This name is used to display the component type in the *Components* task card of SIMIT The name is also used as the basis of the automatic name assignment when instantiating the component type on a chart. The name is independent of the file name under which the component type is stored in the file system.
- Version
  You can set the "Version" of a component type as desired.
- Firm code
- Product code
  The licensing of library components is provided through the "Firm code" and "Product code". If "Lib-ID" is available for existing library components, it is applied as the product code. If the firm code is missing for existing library components, the new firm code is used by default.
- Library family
  You can set the "Library family" of a component type as desired.
- Storage location
- UID
  The "Storage location" of the component type in the file system and its unique identifier "UID" are displayed for informational purposes.

Note

The specifications of "Version" and "Library family" are displayed in the preview of the *Components* task card in SIMIT, but are not evaluated further.

If you change the name or version of a component type, the dialog for file selection will open automatically when saving so that you can also save this component under a new file name. The default file name will match the component name.

7.2.3.3 Protection of the component type

You can assign a password to prevent the component type you have created from being opened in the CTE by unauthorized persons. To do this, simply enter a password. You will be prompted to confirm the password by entering it again.

A password prompt is displayed when you try to open a component type with password protection.

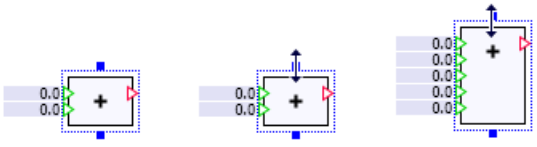
The password protection has no effect on the use of a component type in SIMIT. It can be dragged onto a chart, instantiated and interconnected, just like any other component type.

Note

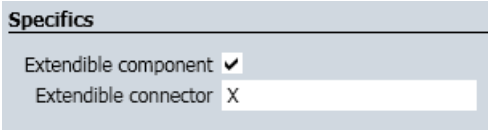
Keep the password in a safe place. You are not be able to open this component type in the Component Type Editor without the right password, even if you have created it.

7.2.3.4 Special properties

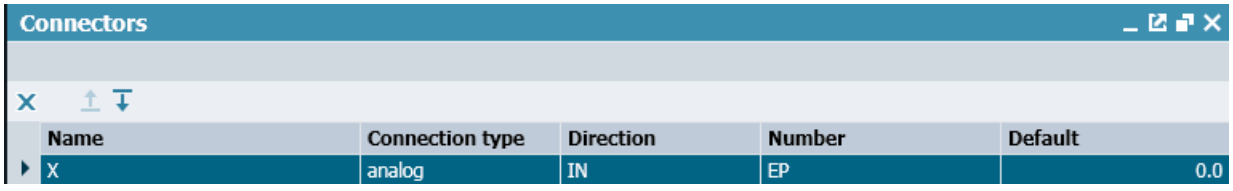
A component type may be assigned the special general property of "graphically scalable". In this case, the component type has exactly one connector (*Graphically scalable connector*) defined as an input or output which can be changed to any number in every instance. The number of connectors is set on the chart by scaling the symbol vertically using its handles on the selection frame.

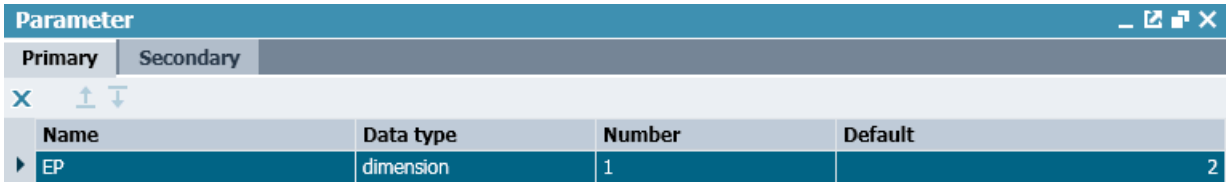


To set this property, use the *Is graphically scalable* option in the editor and specify which connector can be scaled graphically.



The graphically scalable connection must be defined as a vector of connectors of which there is a variable number, whereby the number must be created as a parameter of the type *dimension*.





A component type may have other connectors in addition to the one that is graphically scalable. However, the graphically scalable connector must always be positioned on its symbol below all the other connectors.

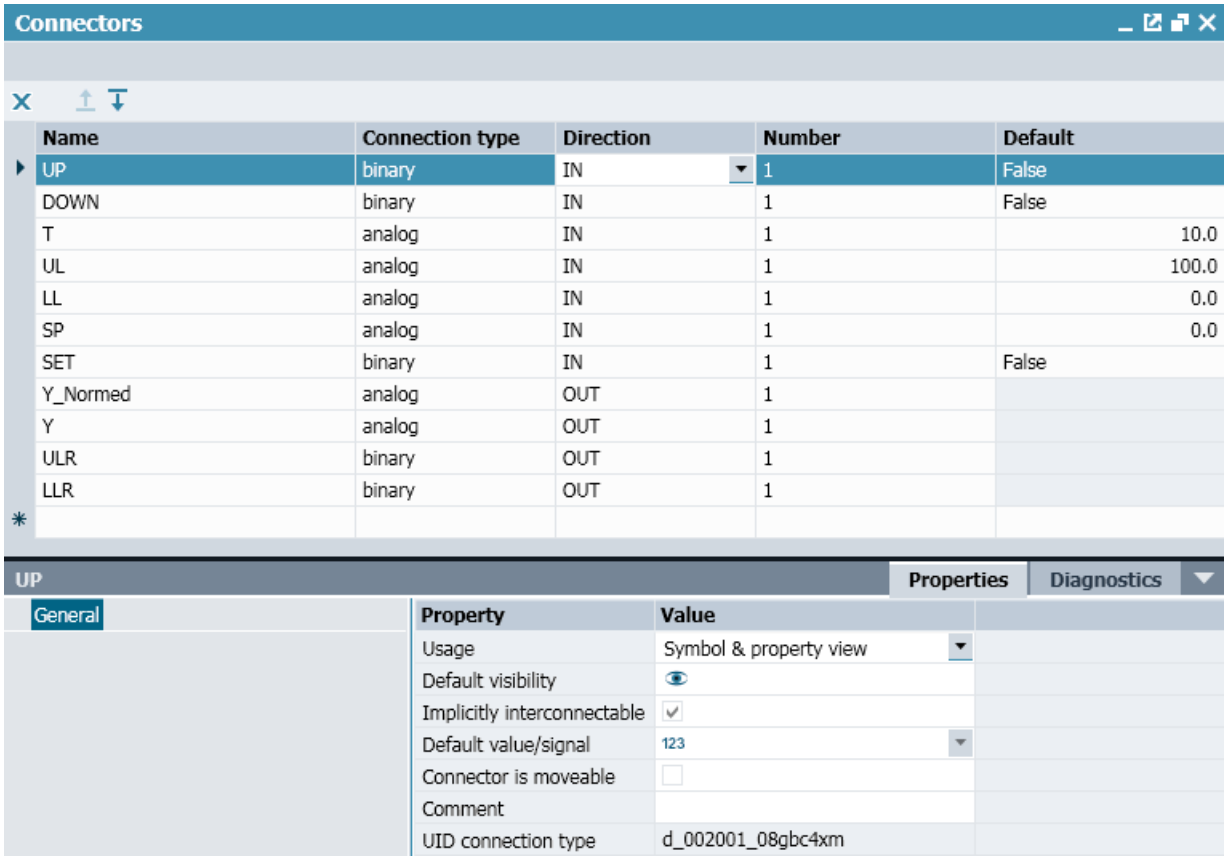
7.2.3.5 Change comments

The change comments in the component type are for documentation purposes only and are not evaluated by SIMIT. You can keep the change history of a component type here, for example.

7.3 Connectors and parameters of component types

7.3.1 The connector editor

The connectors of a component primarily define the interface that is used to exchange information with other components. Connectors are also used to incorporate signals into their operating window. All connectors of a component type are edited in the connector editor, which is created as a table editor. Open the connector editor by double-clicking the "Connectors" menu item in the project tree. The following figure shows the connector editor:



Every connector is identified by the following properties:

- **Name**
  Every connector must have a unique name. The name must contain only letters, digits and the underscore character, and must start with a letter. The name is case sensitive. Reference is made to the name of a connector in the behavior description, for example.
- **Connection type**
  All connectors in SIMIT are typed, which means the connection type precisely defines which information can be exchanged via a connector of this type. Connectors must always be of the same type to be connected to one another on a chart. The available connection types are listed in a selection box.
- **Direction**
  The direction defines whether the connector is defined in the IN or OUT direction. Binary, integer and analog connectors are thus defined as inputs or outputs.
  The special case of a connector without direction (NONE) is only of relevance in association with special libraries. You can find further details in the manuals for these libraries.
- **Number**
  If you have entered a value other than the default number one, you have defined a connector vector with the specified number of elements. These connectors are simply numbered consecutively in the component instance by appending an index number starting with one to the name.
  You can also enter a parameter as number that determines the number of connectors. This parameter must then be of the type *dimension*.
- **Default**
  Connectors that are defined as inputs can be given a default numerical value. This default setting can be overwritten in every component instance.

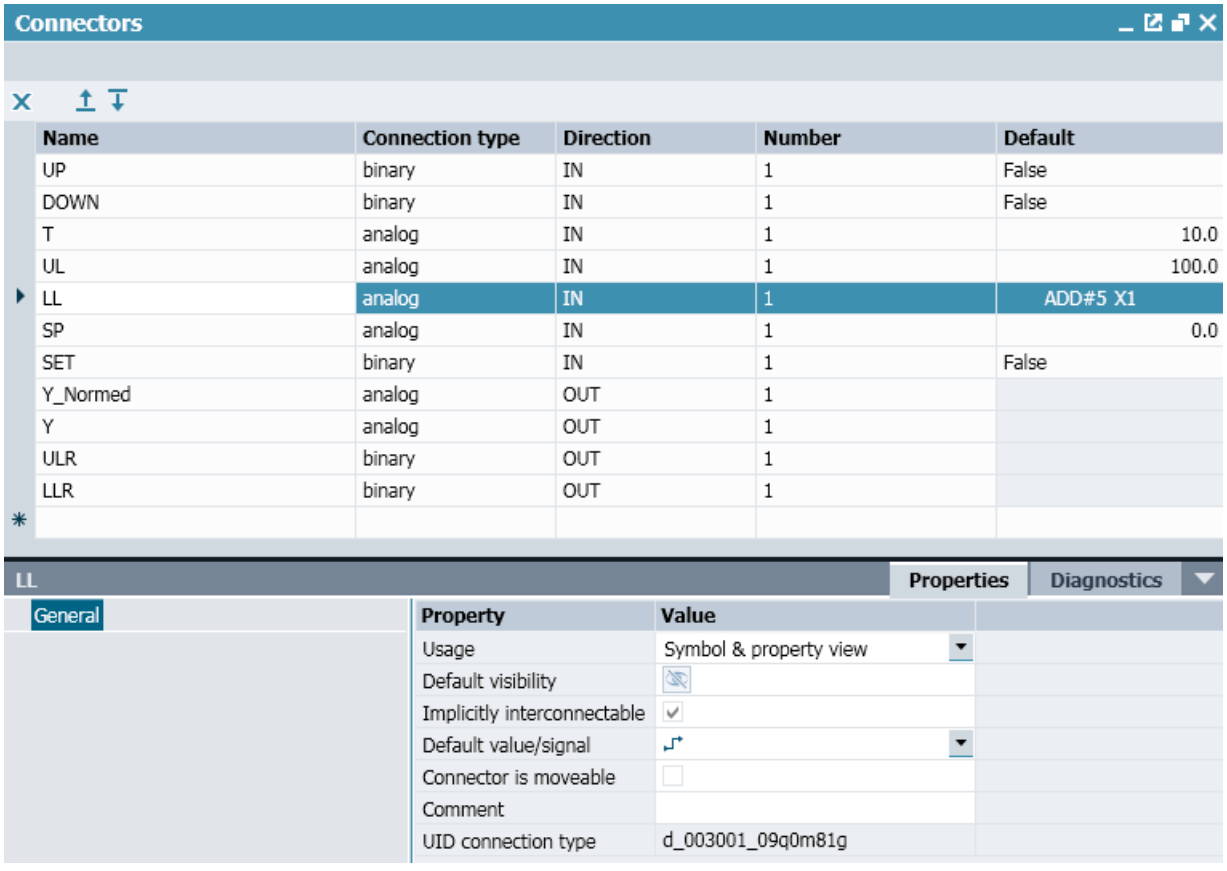
Connectors also have other properties that can be defined in the properties window for every connector:

- **Usage**
  A connector can be used in different ways. The connector should generally be visible on a chart at the component symbol so that it can be interconnected with other components. Set the usage to *Symbol and properties window* to do this.
  If you want the connector to only be visible in the properties window for a component, and not on the component symbol on the chart, set the usage to *Only in property view*. This connector will then be permanently identified as an invisible connector by the symbol in the properties window for the component.
  The *In CTE only* setting allows the connector to be used in the component type, but not to be visible on the symbol or in the component properties window.
- **Default visibility**
  If a connector has the usage *Symbol and properties window*, you can set whether it is initially visible ( ) or invisible ( ) after instantiation on a chart.

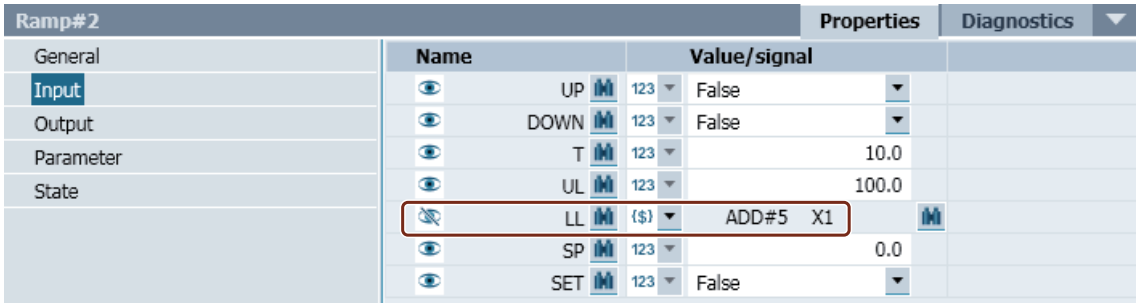
- **Implicitly interconnectable (for input signals only)**
  All connectors that have the usage *Symbol and properties window* can also be implicitly connected in the properties window for the component instance. The *Implicitly interconnectable* property is permanently set for this usage.
  Here you can define whether connectors with the usage *Only in property view* should be implicitly interconnectable connectors or not. If you set a connector with this usage to not be implicitly interconnectable, only the default assignment for this connector can be overwritten in the properties window for the component. If you set it to implicitly interconnectable, the default for Value/Signal is changed to Signal.
- **Default value/signal (for input signals only)**
  If the usage of the connector is set to *Symbol and properties window*, you can select whether the connector is set to *Value* (123) or *Signal* ( ) by default in the component instance.
  If the default is *Signal*, the default visibility setting is automatically set to *Not visible*.
- **Connector is moveable**
  You can define whether the connector in the component instance may be moved on the outer edge of the component. Hold down the "Alt" key and drag the connector with the mouse to move it.
- **Comment**
  The comment for a connector is for documentation purposes only and is not evaluated by SIMIT.
- **Connection type ID**
  Because the name of a connection type does not have to be unique, the unique ID allows you to identify it if there is any doubt.

7.3.2 Special default setting for implicitly interconnectable inputs

Inputs are normally set by default to a numerical value for analog and integer inputs or to the value True/False for binary inputs. If the default setting for *Value/Signal* is set to *Signal* ( ), there is yet another option for the default setting: You can now set a default signal name.



The symbol {$} for this input is then set to the signal name specified in the properties window for an instance of the component. This input of the component is thus permanently interconnected to the output of another component. In the sample signal shown in the figure above, the output *X1* is a component with the name *ADD#5*.

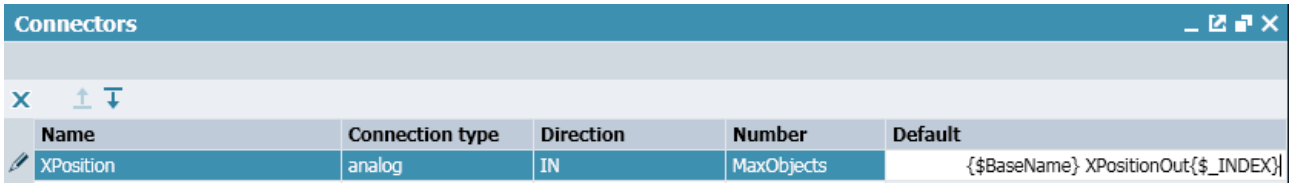


Rather than fixed names for the signal, you can also use parameters or the component instance name. To do this, write the parameter name or *_NAME* for the instance name in curly brackets, preceded by the $ character for the source and/or connector of the signal:

- `{$Parameter name}` or
- `{$_NAME}`.

The parameter name and *_NAME* are thus merely placeholders in the component type for the values of Parameter value or Instance name that are assigned in the component instance. You can also compose the signal name as required from placeholders and fixed names.

You may use the system variable *_INDEX* to define implicit connections for individual elements of a vector. Use {$_INDEX}, as shown in the following figure, for example. When instantiating the component this expression is replaced in each element by the element index, whereby the index counting starts at one.



If the following values have been set for a component as shown in this example,

- Parameter MaxObjects is set to 2 and
- Parameter BaseName is set to "LifterBase#1"

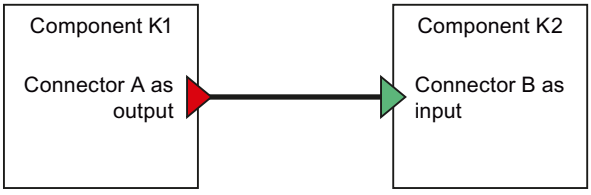
it results in the following default values for the input vector XPosition:

- XPosition1: LifterBase#1 XPositionOut1
- XPosition2: LifterBase#1 XPositionOut2

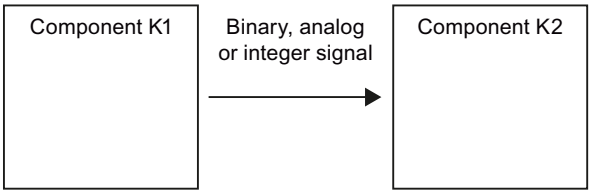
7.3.3 Complex connection types

In the basic connection concept, a single analog, integer or binary signal is transferred between connected connectors of components. The signal connection is always directed from output to input, which means the direction is determined implicitly by the type of connector. This concept for connecting an output to an input is illustrated in the figure below.

Connection view:



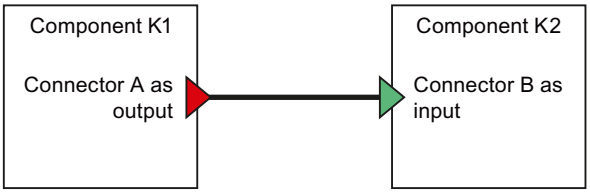
Technical view:



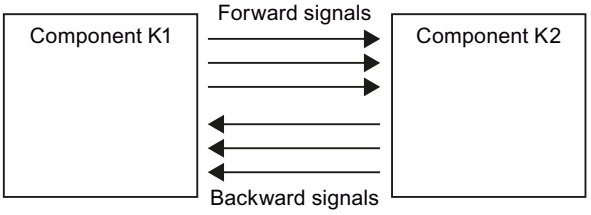
Connections of this type are provided as basic connection types in SIMIT. These types are offered in the selection screen as *analog*, *integer* and *binary*.

The SIMIT connection concept has been extended compared to this basic concept: A connection may be used to transfer multiple signals between connectors in both directions. The direction of a signal thus can no longer be derived from the connected connectors, so it needs to be defined as a *forward* or *backward signal* in the connection type. Forward signals are transferred from an output to an input; backward signals are exactly the opposite. The diagram for such a complex connection is depicted in the figure below.

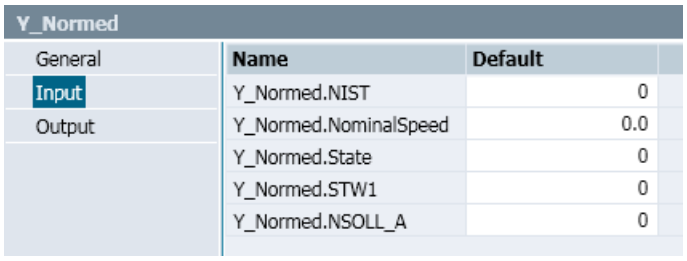
Connection view:



Technical view:

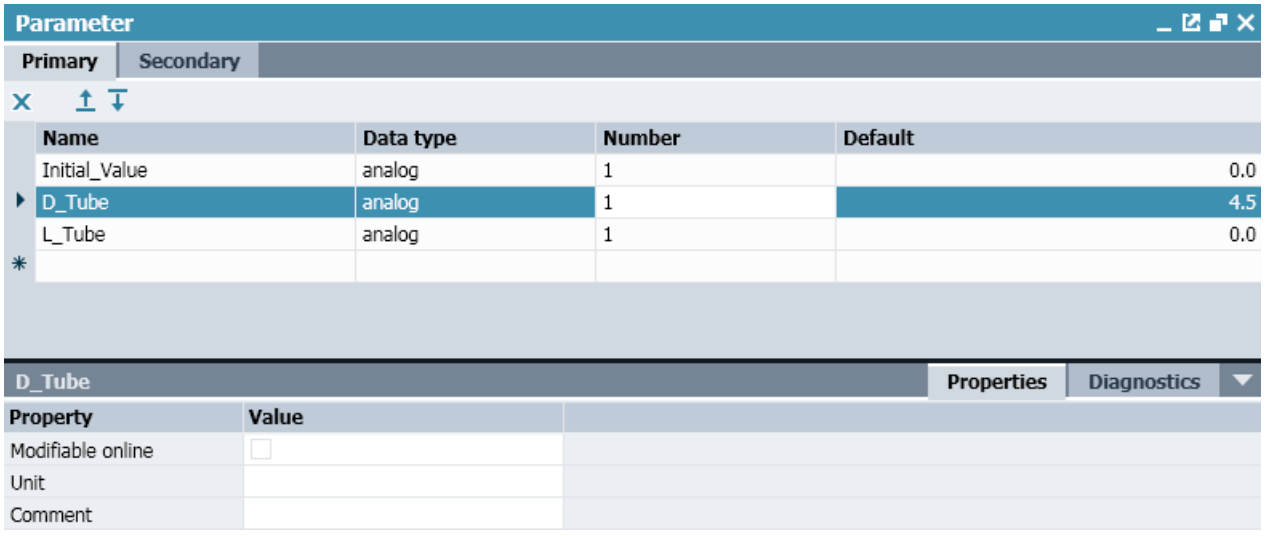


Thus both input and output signals can result from a connector of a complex connection type. These signals are listed in the properties window of the connector, where default inputs can be individually specified.



7.3.4 Parameters of component types

Components can be individually configured using parameters. To do this, the corresponding parameters must be provided in the component type. To define the parameters, open the parameter editor by double-clicking the *Parameter* aspect in the project tree.



Divide the parameters into the "Primary" and "Secondary" tabs. This separates the parameters into "important" and "less important" parameters. Important parameters are used, for example, for assigning the component parameters. If you define secondary parameters here, SIMIT takes this distinction into account in the properties window for the component instance. A further category (*Additional parameters*, which contains the secondary parameters) will then appear in the properties window in addition to the *Parameters* category.

Parameters are identified by the following properties:

- **Name**
  Each parameter must have a unique name. The name must contain only letters, digits and the underscore character, and must start with a letter. The name is case sensitive.
- **Data type**
  Parameters can have one of the data types listed in the table below.
  All enumeration types are also available for parameters. The enumeration types are described in detail in the section "Defining enumeration types (Page 362)".

Table 7-3 Data types for parameters

Data type	Meaning	Value range
binary	Binary values	True/False
analog	Floating point values	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$
integer	Integer values	–9.223.372.036.854.775.808 bis +9.223.372.036.854.775.807
dimension	Number of elements in a connector vector or parameter vector	1 .. 256
text	Single line of text	
characteristic	Characteristic	

- **Number**
  If you have entered a value other than the default number one, you have defined a parameter vector with the specified number of elements. These parameters are simply numbered consecutively in the component instance by appending an index number starting with one to the name.
  You can also enter as the number another parameter that determines the number of this parameter. This parameter must then be of the type *dimension*.
- **Default**
  Parameters can be assigned a default numerical value.

Parameters also have other properties that you can edit in the properties window for that parameter.

- **Modifiable online**
  Modifiable online parameters are parameters that can be changed for a component instance while a simulation is running.
  Parameters of the type *dimension* are not modifiable online.
- **Unit**
  The unit entered here only appears as an additional property of the parameter in the properties window for the component instance.
- **Comment**
  The comment for a parameter is for documentation purposes only and is not evaluated by SIMIT.

All names of parameters and connectors must be unique, which means a connector must not have the same name as a parameter and vice versa.

7.4 The behavior of a component type

7.4.1 Introduction

The functional behavior of a component type is defined by state variables and the behavior description. The behavior description is subdivided into the following aspects:

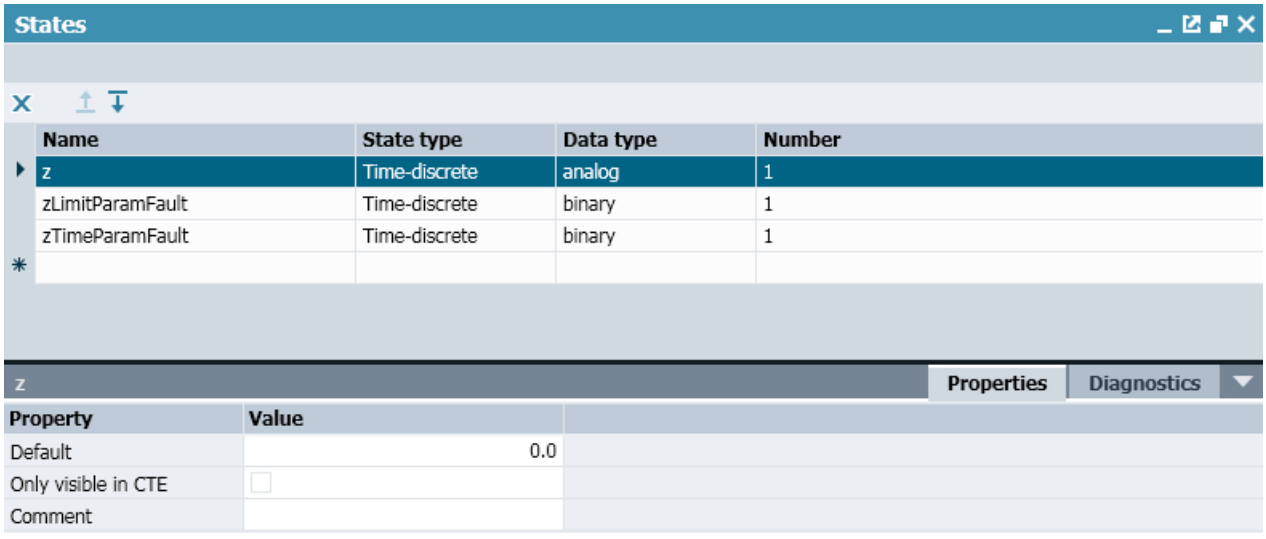
- Initialization
- Cyclic calculation
- Functions

The component type editor provides suitable editors for all aspects: a table editor for the states and a text editor for the sub-aspects of the behavior description.

7.4.2 State variables

State variables of a component are the memory of a component, so to speak. They contain values that, at any point in time, cannot be calculated from the input variables and parameters alone, but depend on what has happened in the past. For example, the fill level in a container cannot be calculated by simply balancing the inflow and outflow at a given point in time; it also depends on the content of the container before the point in time under consideration.

A table editor is provided for editing the states. You can open the editor by double clicking the aspect *States* in the project tree.



A state has the following properties:

- **Name**
  Each state variable must have a unique name. The name must contain only letters, digits and the underscore character, and must start with a letter. The name is case sensitive.
- **State type**
  There are two different state types: *time-discrete* and *continuous*. The difference is determined by how the new value of a state is calculated:
  For time-discrete states, the value is calculated in every processing cycle by a calculation rule in the form of an explicit equation that you define in the behavior description. The rule for calculating a continuous state variable is defined by a differential equation. SIMIT calculates the state values in every processing cycle by solving this differential equation using a suitable numerical method.
  You can find detailed information about how to handle time-discrete and continuous state variables using explicit equations and differential equations in the behavior description in the relevant paragraph in section: The equation-oriented approach (Page 403).
- **Data type**
  Time-discrete state variables can have any of the data types listed in the table below. Continuous state variables are always of the type *analog*.

Table 7-4 Data types for time-discrete states

Data type	Meaning	Value range
binary	Binary values	True/False
analog	Floating point values	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$
integer	Integer values	–9.223.372.036.854.775.808 bis +9.223.372.036.854.775.807
byte	Bytes	0 to 255

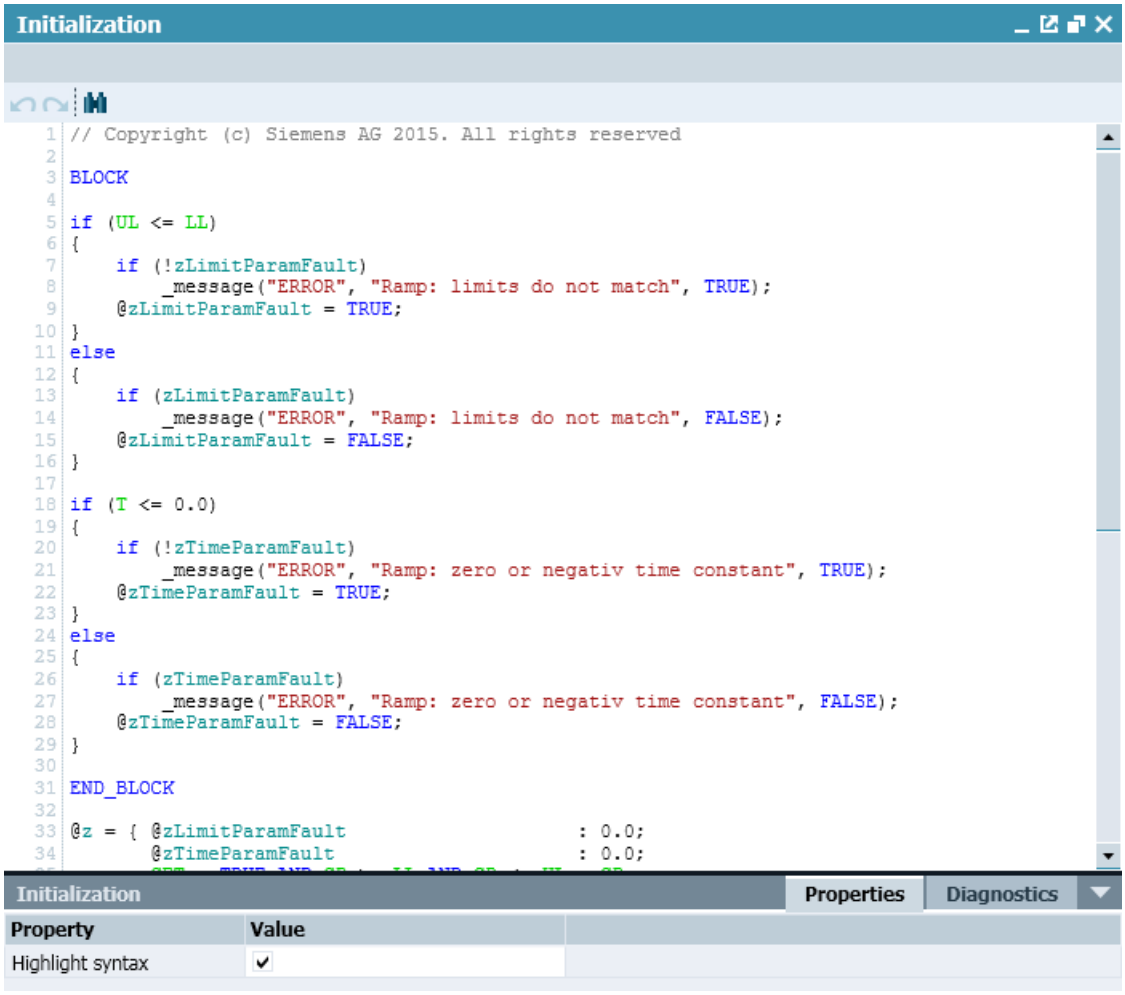
- **Number**
  If you have entered a value other than the default number one, you have defined a state vector with the specified number of elements. These states are simply numbered consecutively in the component instance by appending an index number starting with one to the name.
  You can also enter as the number another parameter that determines the number of this parameter. This parameter must then be of the type *dimension*.

State variables also have other properties that can be defined in the properties window for the component type:

- **Default**
  Every state variable has a default setting matching its type. This default setting can be overwritten in the component instance.
- **Only visible in CTE**
  Set this option if you do not want this state to be visible in the component properties window.
- **Comment**
  The comment for a state variable is for documentation purposes only and is not evaluated by SIMIT.

7.4.3 Initialization, cyclic calculation and functions

The behavior description for a component consists of a part that is executed once during initialization and a part that is executed in every cyclic calculation step. Calculations that are used multiple times can optionally be defined as functions. The Component Type Editor provides a text editor for each of these three sub-aspects of the behavior description. The editor for any of these sub-aspects can be opened by double-clicking the corresponding sub-aspect in the project tree.



For ease of orientation, you can activate the *Highlight syntax* option in the properties window. Important elements of the description syntax are then made easier to identify by different text colors. The table below lists the colors used for the individual elements.

Table 7-5 Colors used for the elements

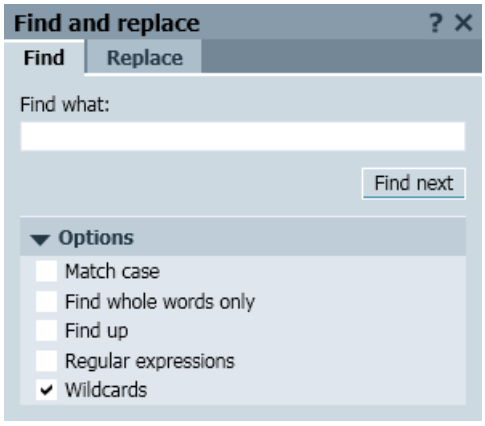
Element	Color
Input signal	Green
Output signal	Red
State	Olive green
Parameters	Pink

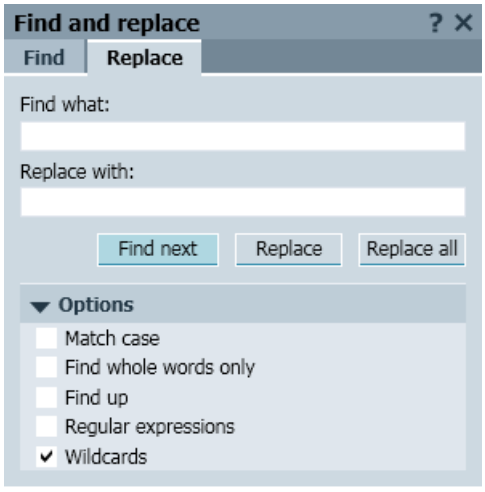
Element	Color
Text constant	Brown
Keyword	Blue
Comment	Gray

Note

The use of highlight colors increases the computing power needed to update the user interface of the text editor; the extra time needed is sometimes sufficient to cause a short delay while you are typing. You may therefore find it useful to switch off the highlighting, at least temporarily, if your texts are very long.

All three text editors offer a function for finding and replacing texts under the icon. You can also call up this function using the *Ctrl+F* or *Ctrl+H* shortcuts. There are various options you can use for searching, as shown in the figure below.





You can find additional information on behavior description in the section Syntax for the behavior description (Page 393).

7.4.4 Data record handler

Data record handler

The data record handler is an editor in which you program the response behavior of a component to acyclic data record requests. In acyclic data record communication, the initiative always comes from the controller. The component responds to the requests according to the programming.

If no component responds to the data record, the coupling responds automatically.

This mechanism is supported by the following couplings:

- SIMIT Unit
- SIMIT Virtual Controller
- PLCSIM Advanced

Example of acyclic data record communication

Products are to be transported to different plant areas based on printed QR codes in a plant. A camera scans the QR codes on the products for this. When the controller sends a data record write request to the camera, the data is to be transferred.

To simulate this camera, you need to create a new component and program the response to the data record request there.

Response to data record requests

You program the response to a data record request in the "Data Record Handler" editor of the component type editor. Use the methods WRREC_Handler and RDREC_Handler.

Method	WRREC_Handler	RDREC_Handler
Application	Response to data record write requests of the controller	Response to data record read requests of the controller
Syntax	WRREC_HANDLER (Unit [, data record number]) ... END_HANDLER	RDREC_HANDLER (Unit [, data record number]) ... END_HANDLER

Note

If the data record number is specified, the method is executed only when the controller wants to read or write that exact data record. If no data record number is specified, there is a response to all read and write requests.

The following components are examples of data record handlers:

ReadRecordHandler – Response to a read request of the controller (Page 578)

WriteRecordHandler – Response to a write request from the controller (Page 578)

Connectors and parameters

You have to create a `Unit` transfer parameter at the component. At the `Unit` parameter, connect the unit connector that identifies the connected module.

The following variables are important for the handler methods:

- **Data record number "_DR_number"**
  Data record number for use in programming the component.
- **Data record length "_DR_length"**
  Indicates the length of a data record.

- **Byte array "_DR_data"**
  Contains the data transferred/read by the controller as a byte array.

  Note

  The byte array "*_DR_data*" has the dimension "*_DR_length*". If the controller wants to access a larger data range, the system may demonstrate unpredictable behavior or it may crash entirely.

- **Return value "DR_result"**
  The component must return a return value. Return values below 0x8000 are interpreted by the controller as the size of the data record. If the return value is above 0x7FFF, the control interprets the return value as an error message.
  The following error codes are supported by all couplings:
  – 0x80B0 Data record unknown
  – 0x80B1 Length error of data record

  Other error codes cannot be supported by all couplings. These other error codes are passed on to the coupling partner, but are also recorded in the tracer as "Warning".

See also

Unit connector (Page 437)

Access to a data record or memory area (Page 216)

7.4.5 Topology

The topology description is only meaningful in association with the libraries CONTEC and FLOWNET.

For more information on topological aspects, refer to the section Topological properties (Page 945) The CONTEC library and in the section Topological properties (Page 852) The FLOWNET library.

7.5 Visualization of component types

7.5.1 Introduction

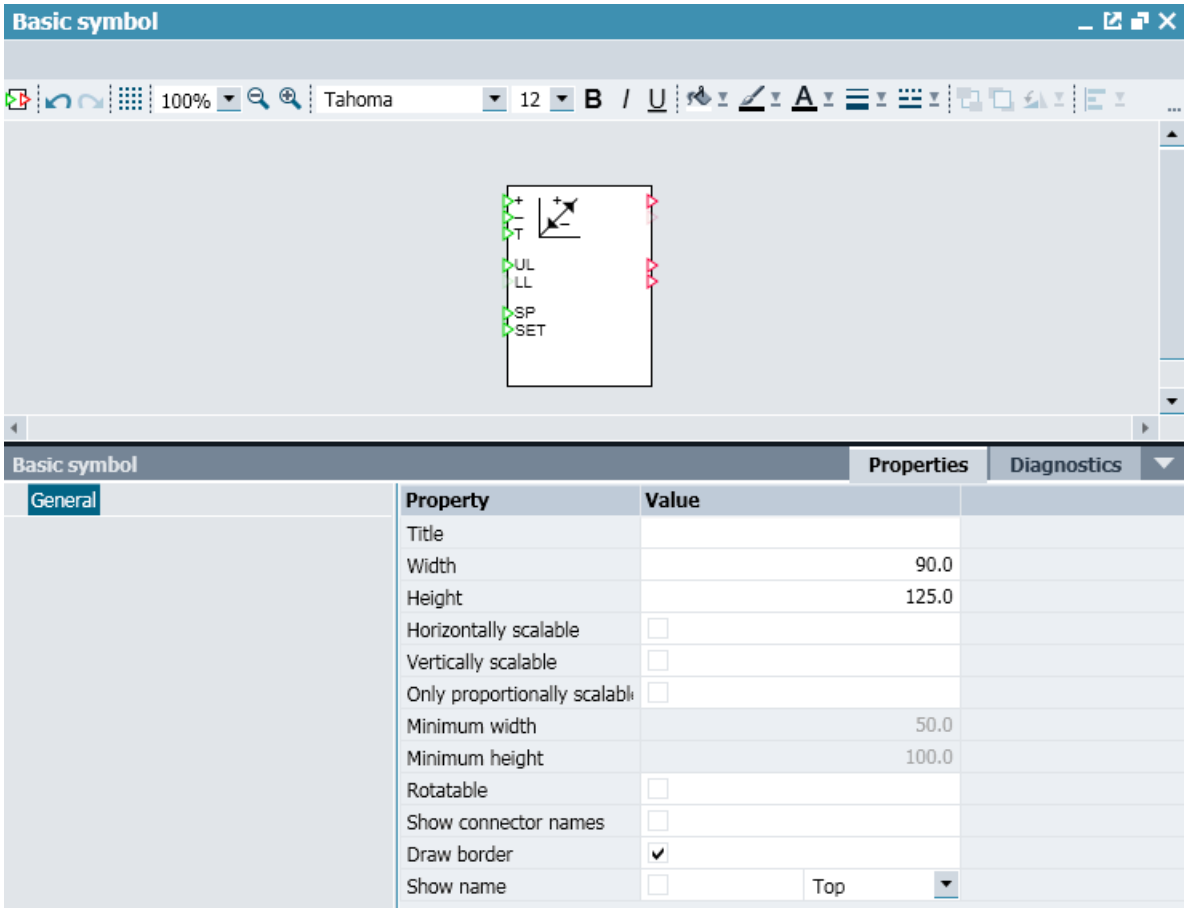
A graphical representation of a component instance is created in the form of a basic symbol for every component type. The basic symbol is displayed in the preview of the *Components* task card. It represents every component instance on the chart.

Optionally a link view can be created for a component type. This provides additional access to the component instance. It is also possible to define an operating window for a component type that can be opened for every instance while the simulation is running to set and display component values.

7.5.2 The basic symbol

7.5.2.1 Editing the basic symbol

To edit the basic symbol, open the graphical symbol editor by double-clicking the *Basic symbol* aspect in the project tree. The *Graphic* task card contains the graphical elements of the chart editor for designing the graphical aspects of the basic symbol. You can use these graphical functions to design the basic symbol as required in the available space.



You can apply animations to graphical elements used for creating the basic and link symbol.

7.5.2.2 Editing graphics

You are provided the full set of features available that the graphics editor in SIMIT for editing graphics. You can find more information about this topic under "Visualizing graphics (Page 254)".

7.5.2.3 Editing connectors

All connectors as specified in the connector editor can be automatically arranged on the basic symbol. Just click the symbol in the toolbar. Connectors are automatically arranged in the specified order along the border of the basic symbol: Inputs are placed on the left hand side, outputs on the right hand side of the symbol. You may use the mouse to drag any connector to the desired position within the predefined grid resolution of 5 pixels.

The coordinates of a connector are displayed in the properties window. You can enter the desired position there manually. In contrast to positioning with the mouse, manual input is not limited to the grid resolution. Coordinates therefore also do not need to be integer values.



If there is a vector of inputs or outputs that has a fixed, which means not variable dimension, each individual connector of that vector can be freely positioned.

Note

Changes in the connector editor do not appear in the symbol editor until you save the component type or run an update from the toolbar ( ) or by pressing function key F5.

7.5.2.4 Editing properties

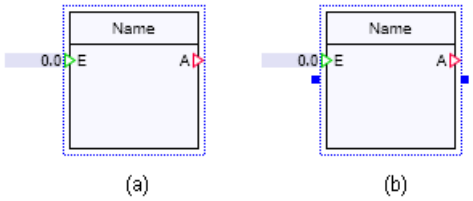
You can define properties for the basic symbol in the properties window:

- **Title**
  If you enter a *Title*, this appears centered at the top of the basic symbol and is offset from the rest of the area by a horizontal separating line.
  When combined with the *Draw border* option, this represents a rudimentary way to design a component view.



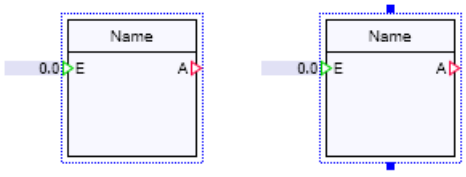
- **Width**
  Here you can specify the *Width* of the basic symbol in pixels as a numerical value. You can also move the left or right edge of the symbol area in the editor window by holding down the left mouse button. The numerical value for the width is automatically corrected.
- **Height**
  Here you can specify the *Height* of the basic symbol in pixels as a numerical value. You can also hold down the left mouse button on the top or bottom edge of the symbol area in the editor window to move it. The numerical value for the height will change automatically as you do so.

- **Horizontally scalable**
  The *Horizontally scalable* option is used to define whether the basic symbol for the component instance should be scalable in the horizontal direction on a chart. Suitable handles are then provided on the selection frame of the basic symbol.



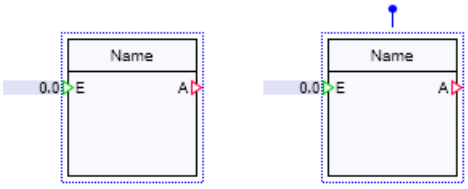
(a) (b)

- **Vertically scalable**
  The *Vertically scalable* option is used to define whether the basic symbol for the component instance should be scalable in the vertical direction on a chart. Suitable handles are then provided on the selection frame of the basic symbol.



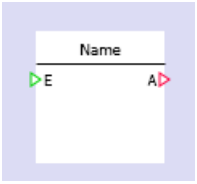
(a) (b)

- **Minimum width**
  The *Minimum width* of the basic symbol cannot be undershot when scaling the basic symbol horizontally on the chart.
- **Minimum height**
  The *Minimum height* of the basic symbol cannot be undershot when scaling the basic symbol vertically on the chart.
- **Is rotatable**
  This option defines whether the basic symbol can be rotated on a chart or not. If it is rotatable, a suitable handle appears on the selection frame of the basic symbol.

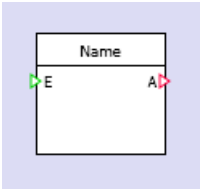


(a) (b)

- **Show connector names**
  This option allows you to define whether the connector names of the inputs and outputs should be displayed in the symbol. Note that inputs can only be displayed on the left edge and outputs on the right edge of the basic symbol.



- **Draw border**
  This option allows you to define whether the basic symbol should appear with a black border.



- **Only proportionally scalable**
  This option can be used to specify that a component width and height cannot be scaled independently but only proportionally, which means maintaining a constant ratio between width and height. This option can only be selected when the component is both horizontally and vertically scalable.
- **Scale**
  When creating user types for conveyor technology components (components of types from the CONTEC library) you may specify a scale for the basic symbol. This scale fulfills two tasks:
  - All sizes and positions can be defined in millimeters according to the scale specified when creating the symbol of a handling equipment component.
  - When creating the symbol of an object component, the size resulting from the specified scale is the default size with which the object is placed in the material list in SIMIT.

Note

The *Scale* property is only available if you have licensed the CONTEC library in SIMIT.

7.5.2.5 Scalability

When creating your own conveyor technology component types you can also set a scale for the basic symbol and the link in the CTE. This scale fulfills two purposes:

1. When modelling handling equipment, all dimensions and positions can be specified in millimeters relative to the selected scale.
2. When modeling an object, the dimension which results from the selected scale in the CTE is the predefined size with which the object is created in the material list.

Note

This function is only available if the CONTEC library has been licensed.

The following system variables permit access to the actual dimensions of a component in the behavior description:

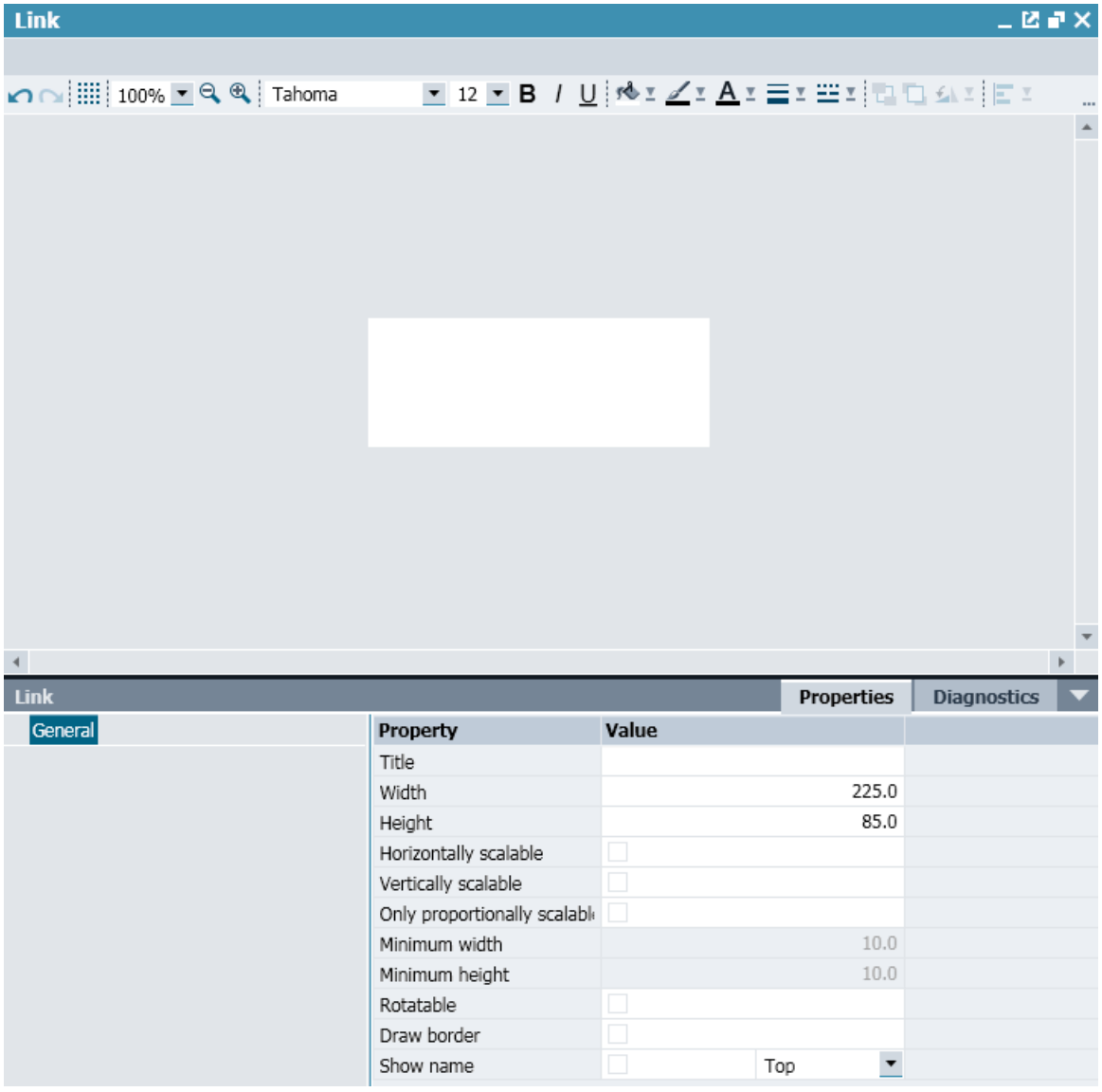
Table 7-6 **System variables to determine component dimensions**

System variable	Data type	Meaning
_WIDTH	analog	Width (unscaled) of the component in pixels
_HEIGHT	analog	Height (unscaled) of the component in pixels
_SCALEX	analog	Horizontal scaling of the component
_SCALEY	analog	Vertical scaling of the component
_TECHSCALE	analog	Scale of the chart on which the component is placed (in mm per pixel)

7.5.3 The link symbol

The basic symbol of a component type is used to parameterize and connect the component instance on a chart. Optionally, a component type can have a link view. The symbol shown in this view – the link symbol – may have a graphic design that is completely independent of the basic symbol. Unlike the basic symbol it has no connectors. Otherwise, the graphic elements of the link symbol can be freely designed as in the basic symbol. Thus, they can be animated, for example, to visualize current simulation states of a component instance.

Double-click the *Link* entry in the project tree to open the editor for the link symbol. The graphical editor that opens provides the same functions as the basic symbol editor, except for the functions that relate to connectors.

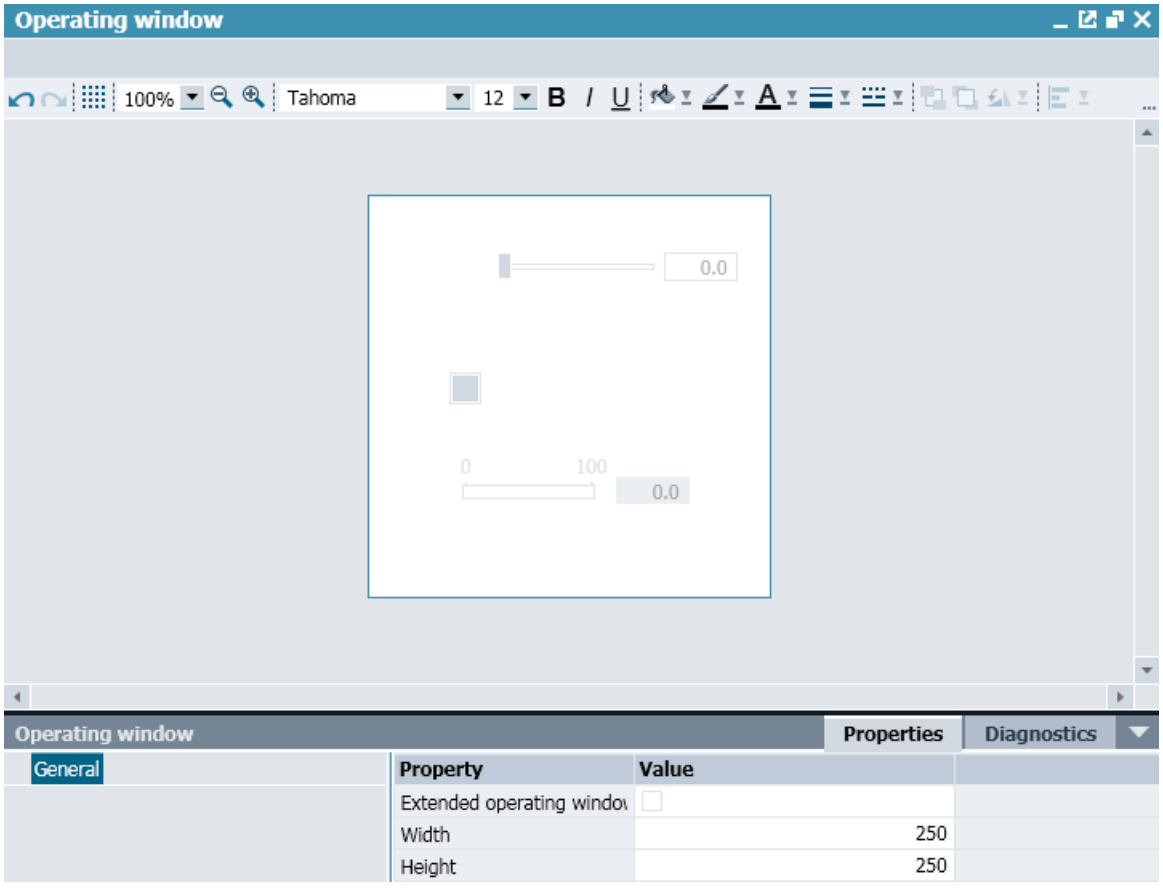


You can find more information on use of the link view in section Creating and editing charts (Page 252).

7.5.4 The operating window

An operating window allows you to set and display values of the component instance while the simulation is running. Double-click the component in the chart to open the operating window.

To create an operating window, open the editor by double-clicking the *Operating window* entry in the project tree.

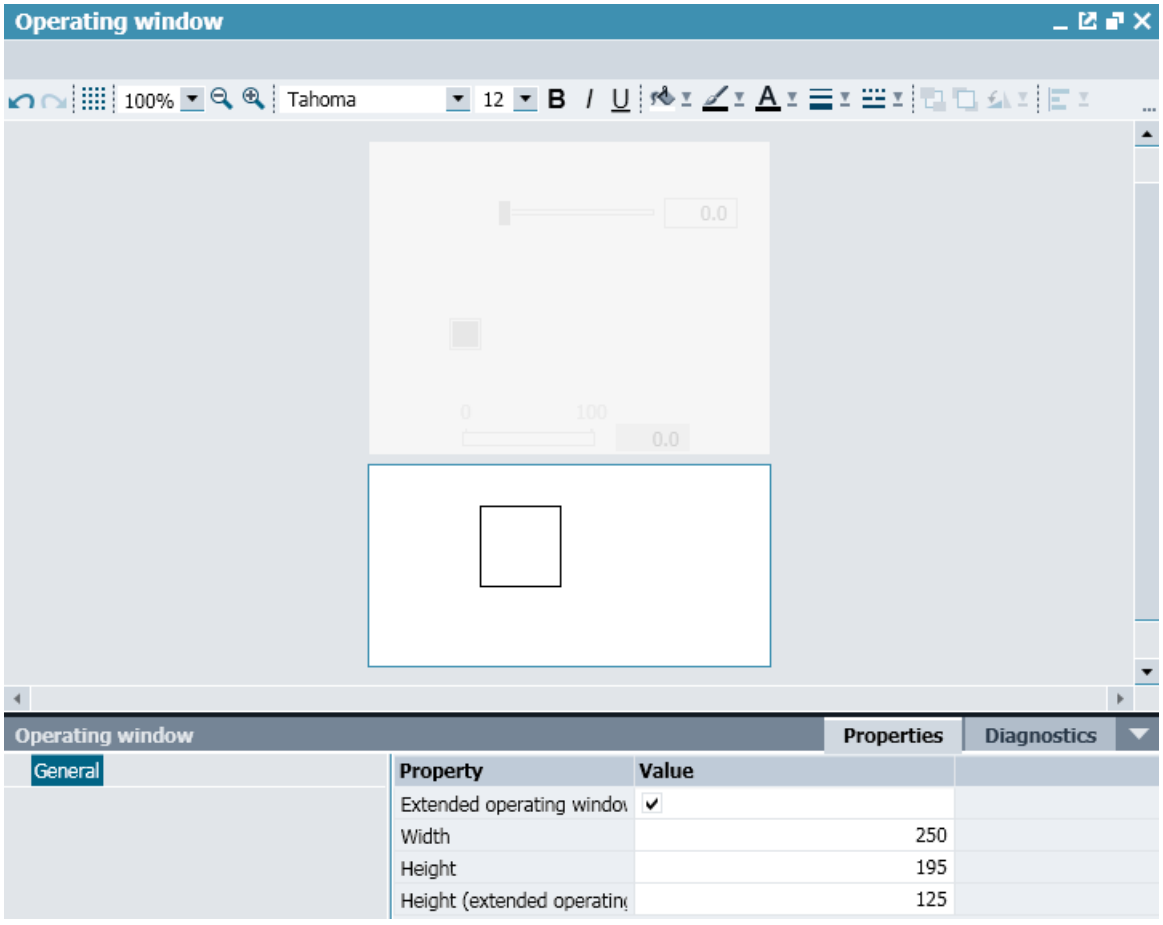


You can use all controls for the SIMIT basic library in an operating window. These controls are provided on the *Controls* task card. You can drag these controls from the task card onto the drawing surface of the editor and connect them to suitable input or output signals of the component type.

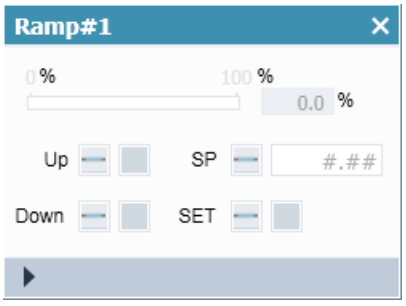
The *Graphic* task card also provides the SIMIT graphic functions for graphically designing the operating window. Note that graphic objects in the operating window cannot be animated.

The properties window contains further options for designing the operating window:

- **Extended operating window**
  You can divide the operating window into two areas, for example, to separate frequently used operator controls from those that are needed less often. To do this, activate the *Extended operating window* option. The editor then contains another area for the extended operating window. This area is just as wide as the area for the operating window, and the height can be changed as required.



In the open operating window for a component instance, you can easily open the extended operating window by clicking the bottom edge.



- **Width**
  Here you can specify the width of the operating window in pixels as a numerical value. You can also move the left or right edge of the window area in the editor by holding down the left mouse button. The numerical value is updated automatically.
- **Height**
  Here you can specify the height of the operating window in pixels as a numerical value. You can also move the top or bottom edge of the window area in the editor by holding down the left mouse button. The numerical value is updated automatically.

7.6 Documentation of component types

You can add a description to a component type in the "Documentation" subfolder, which will then become part of this component type. This description can then be called up in the case of use on a chart or in the Components task card.

Two different descriptions can be inserted, which will then be used depending on the language setting in SIMIT. To do this, use the "Select" shortcut menu of the "German" or "English" subfolder.

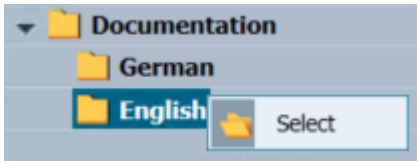


Figure 7-1 Documentation of component types

Only documents in .mht format ("MIME Encapsulation of Aggregate HTML") can be used. For example, you create this type of document with Microsoft Word or save existing Word documents in this format. Word will designate the document as a "Single File Web Page".

You can open documents embedded in this way in the component type editor for inspection or remove them again.

If another document is selected, it will replace an existing document.

7.7 Syntax for the behavior description

7.7.1 Overview

Note

The "equation-oriented approach" is obsolete and should not be used anymore, because it will no longer be supported in future versions!

The behavior description for a component consists of a part that is executed once during initialization and a part that is executed in every cyclic calculation step. The same description syntax applies to both parts.

The behavior description is made in one or more blocks in which the behavior of the component is formulated in an instruction-oriented language. The order in which these blocks are executed during the simulation is automatically determined by SIMIT based on the input and output variables used. The instructions are executed in the given order within a block.

It is often not necessary to use multiple blocks. However, when using differential equations for example, it is advantageous to write the differential equations in a separate block. See "Differential equation systems (Page 402)".

The formerly used "equation-oriented approach" is outdated and should not be used anymore, because it will not be supported in future versions.

7.7.2 Conversion of the behavior description to C# code

The behavior description in the simulation project is converted into C# code for each component instance. This conversion takes place automatically and cannot be seen by and is of no significance to the user.

Since not all syntax errors can be detected before the code is generated, the C# compiler may also generate error messages. Note that the information in such error messages relates to the generated code that can differ from the behavior specified in the component type in terms of both syntax and breakdown.

7.7.3 The instruction-oriented approach

7.7.3.1 Introduction

The instruction-oriented approach is used to formulate user functions or entire function blocks. No equations are used within user-defined functions and blocks; these are replaced by program commands that are executed in the defined order. The syntax is identical within the function or block definition.

7.7.3.2 Functions

Functions are created in the *Functions* sub-aspect of the behavior description. Double-click the *Functions* entry in the navigation bar to open the text editor provided.

A function consists of the following elements:

- The keyword *FUNCTION*,
- The function declaration
- The instruction part and
- The keyword *END_FUNCTION*.

Syntax:

```
FUNCTION Function_name (Data_type Output_variable1 [, Data_type
    Output_variable2]) : (Data_type Input_variable1 [, Data_type
    Input_variable2])
    Instruction_part
END_FUNCTION
```

Only the basic data types *binary*, *integer* and *analog* are permitted as data types for the input and output variables.

Example:

```
FUNCTION Trigonometry (analog y1, analog y2) :
                (analog x1, analog x2, binary state)
  if (state) {
      y1 = _sin(x1);
      y2 = _tan(x2);
  }
  else {
   y1 = 0.0;
   y2 = 0.0;
  }
END_FUNCTION
```

This function can then be called as follows in the initialization part or in the cyclic calculation:

```
(Out1, Out2) = Trigonometry(In1, In2, In3);
```

7.7.3.3 Blocks

You can add as many blocks as you like to the initialization or cyclic calculation and use an instruction-oriented behavior description within these blocks as with functions.

A block consists of the following elements:

- The keyword *BLOCK*,
- The instruction part and
- The keyword *END_BLOCK*.

Syntax:

```
BLOCK
    Instruction_part
END_BLOCK
```

The instructions within a block are carried out in exactly the same order that you defined them. The order in which the blocks are calculated as a whole, however, is determined by SIMIT based on dependencies in the same way as for the equation-based approach. SIMIT analyzes which variables of the behavior description are calculated within a block, that is, which are changed and to which there is only read access.

If you change a variable in more than one block, for example, by setting an output variable in two blocks, this creates a contradiction that SIMIT is unable to resolve. Multiple variable definitions of this type are therefore not permitted.

7.7.3.4 Local variables

You can define local variables within the behavior description for initialization or cyclic calculation. These can then be used in all blocks. You can find additional information on this in section:Local variables (Page 404). But you can also define local variables within a function or a block:

```
Data_type Name[,Name];
```

Example:

```
binary b1, b2, b3;
```

These variables are then only valid within the block or the function in which they are defined. The data types listed in the table below are permitted for these variables.

Table 7-7 Data types for variables in blocks and functions

Data type	Meaning	Value range	Default
binary, bool	Binary values	True/False	False
analog, double	Floating point values	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$	0.0
integer, long	Integer values	–9.223.372.036.854.775.808 bis 9.223.372.036.854.775.807	0
byte	Integer values	0 to 255	0
sbyte	Integer values	–128 to 127	0
ushort	Integer values	0 to 65535	0
short	Integer values	–32.768 to 32.767	0
uint	Integer values	0 to 4,294,967,295	0
int	Integer values	–2.147.483.648 to 2.147.483.647	0
ulong	Integer values	0 to 18,446,744,073,709,551,615	0
text, string	Character string (text)		

The name of a local variable must contain only letters, digits and the underscore character, and must start with a letter.

Here again the local variables in the block are only used to save interim results that will be needed again in the same processing step. If you wish to access calculated values again in the next calculation step, always create time-discrete state variables rather than local variables.

7.7.3.5 Fields

You can also define local variables as fields (arrays):

```
Data_type Name[Dimension];
```

The dimension must be an integer constant. You can also define a field with values:

```
Data_type Name[] = {Value1, Value2, …};
```

In this case, the dimension is automatically obtained from the number of defined values. The field elements are initialized with the values specified in the definition.

7.7.3.6 Constants

The constants you can use will depend on the data type of the result variable. The constants for each data type are described in the table below.

Data type	Constants
Binary value	"FALSE" or "TRUE"
Floating point value	Decimal fraction with a dot as the decimal mark, for example "125.61" Exponential notation, for example "62.2e-4"
Integer value	Sequence of digits without thousands separator, for example "125985"

7.7.3.7 Loops

DO loop

Syntax:

```
do
        Instruction_list;
while (Condition);
```

Example:

```
i = 0;
do
{
        i++;
        v[i] = 5 * i;
}
while (i < 10)
```

FOR loop

Syntax:

```
for (Initialization; Termination_condition; Iteration)
        Instruction list
```

You can assign initial values to variables in the initialization part. The variables must first have been declared. The termination condition determines how many passes are made through the loops. The iteration is always carried out at the end of the loop.

Example:

```
for (i = 0; i < 10; i++)
    k = k + 1;
```

WHILE loop

Syntax:

```
while (Condition)
        Instruction_list;
```

The instructions in the while loop are executed for as long as the condition is true.

Example:
```
i = 0;
while (i < 10)
{
        k++
        i++;
}
```

7.7.3.8 Conditional branches

IF instruction

The IF instruction has the following syntax:
```
if (Condition)
        Instruction_list;
else
        Instruction_list;
```

The *else* branch is optional and is executed if the condition proves to be false.

SWITCH instruction

The SWITCH instruction is used to compare a variable with different constants.

Syntax:
```
switch (Variable)
{
    case Constant1:
            Instruction_list;
            break;
    case Constant2:
            Instruction_list;
            break;
    [default:
            Instruction_list;
            break;]
}
```

You can create any number of *case* blocks. The keyword *break* is needed and acts as an end marker for instruction lists. The *default* block is optional.

Example:
```
switch (Status)
{
    case 0:
            Output = Input;
            break;
    case 1:
            Output = Factor * Input;
            break;
    case 2:
            Output = -1 * Factor * Input;
```

```
            break;
    default:
            Output = 0.0;
            break;
}
```

The variable that is looked up in the *switch* instruction must be either an integer or an enumeration type. If it is an enumeration type, the alternatives in the *case* instruction are given in single quotes.

7.7.3.9 System functions

You can access the system functions in blocks. Prefix an underscore "_" when calling a system function, for example "`_resetSimTime()`".

The following system functions are available:

- `message(C, T, D)`
  The text `T` is entered in the message system as a message of category `C`. The parameter `D` specifies whether the message is incoming (`D=True`) or outgoing (`D=False`).
  The category and message text must be simple, non-composite text constants.
  Example: `_message("ERROR", "error message", True);`
- `printlog(T)`
  The text `T` is written to a file opened by the script.
  Text parameters, input variables and status variables can also be used in this function. In this case, the current values at the time of the call are transferred.
  Example: `_printlog("calculated value: " + Z + "seconds.");`
- `resetSimTime()`
  The simulation time is reset to zero.

  Note

  Resetting the simulation time can lead to undesired results if simulation mechanisms are directly dependent on the simulation time, such as trend display or isochronous mode.

- `Tools.StringLength(S)`
  Returns the length of the text variable `S` as an integer.
- `Tools.Substring(S, I, L)`
  Returns part of the text variable `S`, starting from position `I` and length `L`.
- `Tools.CharAt(S, I)`
  Returns the character of the text variable `S` at position `I`. The return value is of the type `char`. To receive the numeric code of the character, use the `_trunc2byte()` function.
- `Tools.ConvertToInteger(S, I, L, B)`
  Returns the numerical value that is represented by the text variable `S`; only the part from position `I` with length `L` is included. `B` specifies the number base in which this number is coded (2, 8, 10 or 16).

- `Tools.ConvertToInteger(S, B)`
  Returns the numerical value that is represented by the text variable `S`. `B` specifies the number base in which this number is coded (2, 8, 10 or 16).
- `Tools.Replace` (string input, string pattern, string replacement)
  Returns the modified string `input` in which the regular expression `pattern` was replaced with `replacement` .

You can find additional functions in the section: Function calls for mathematical standard functions (Page 409)

7.7.3.10 Operators

The following operators are available within the functions and blocks. In the table below they are arranged in order from highest to lowest priority. Where several operators appear together in a section, they have the same priority.

Table 7-8 Operators

Operation	Notation
Parentheses	(expression)
Function call	Function name (parameter list)
Increment	I++
Decrement	I––
Increment	++I
Decrement	––I
Unary plus (sign)	+Z
Arithmetic negation	–Z
Bitwise complement	~I
Logical negation	!B
Multiplication	Z * Z
Division	Z / Z
Modulo remainder	Z % Z
Addition	Z + Z
Subtraction	Z – Z
Bitwise left shift	I << J
Bitwise right shift	I >> J
Less than comparison	Z < Z
Greater than comparison	Z > Z
Less than or equal to comparison	Z <= Z
Greater than or equal to comparison	Z >= Z
Equality	Z == Z or B == B
Inequality	Z != Z or B != B
Bitwise AND	I & I
Bitwise exclusive OR	I ^ I
Bitwise inclusive OR	I | I
Logical AND	B && B

Operation	Notation
Logical OR	B || B
Conditional expression	B ? Expression : Expression
Simple assignment	B = B or Z = Z
Composite assignments	Z *= Z Z /= Z Z += Z Z –= Z I <<= J I >>= J I &= I I ^= I I |= I

The type of permitted operand is divided into the listed groups of the table below:

Table 7-9 Operand data types

Title	Data types
B	binary, bool
I	integer, sbyte, byte, short, ushort, int, uint, long, ulong
J	sbyte, byte, short, ushort, int
Z	analog, float, double, integer, sbyte, byte, short, ushort, int, uint, long, ulong

For all assignments it should be noted that it is not possible to assign from a larger to a smaller data type. The table below lists the assignments that are permitted.

Table 7-10 Data type conversion in assignments

Target data type	Permitted assignment data types
binary, bool	binary, bool
sbyte	sbyte
byte	byte
short	short, byte, sbyte
ushort	ushort, byte
int	int, short, ushort, sbyte, byte
uint	uint, ushort, byte
integer, long	integer, long, int, uint, short, ushort, sbyte, byte
ulong	ulong, uint, ushort, byte
float	float, integer, long, ulong, int, uint, short, ushort, sbyte, byte
analog, double	analog, double, float, integer, long, ulong, int, uint, short, ushort, sbyte, byte

There are functions for rounding or truncating the integer component when converting the data type. You can find additional information in the section:Function calls for mathematical standard functions (Page 409).

7.7.3.11 Accessing state variables

You cannot access state variables from functions. You must pass state variables in the function call, if necessary. Within blocks you can access only discrete state variables. The notation shown in the table below must be used for this access.

Table 7-11 Accessing discrete state variables

Notation	Access
Z	Value from the previous computation cycle
@Z	New value

7.7.4 Differential equation systems

7.7.4.1 Introduction

If you defined analog states of your component type as *continuous*, the value that this state is to assume is not specified in the behavior description. Rather, you formulate the derivative of the state with respect to time as a function of the variables and the time variation of this state.

If you define several continuous states within a component, a system of differential equations is automatically created from it. However, differential equation systems are only calculated component by component, dependencies on states of other components are not included in this solver.

SIMIT uses a numerical solver based on the Runge-Kutta-Merson method to solve these ordinary differential equations. Since this method works in several steps, i.e. it calculates the derivatives in a time slice four times with different arguments, it is recommended that all calculations that are directly related to continuous states be combined in a separate block, to prevent having to execute more component code than is necessary.

7.7.4.2 Notation for the differential

The mathematical notation for the differential d/dt is abbreviated with the dollar sign in SIMIT. For example, for the change of the continuous state "Mass" in a container, you write an equation in the form:

```
$Mass = Inflow - Outflow;
```

The new value of the state variable is now determined by SIMIT in every step of the computation by numerically solving the differential equation.

You can also describe multiple continuous states in your component type by differential equations that are dependent on one another. This then yields, for example, the following system of differential equations:

```
$Z1 = Z2 + Factor1 * Z3 – Input1;
$Z2 = Factor2 * Z1;
$Z3 = Z1 + Z2;
```

7.7.4.3 Corrections for the state variables

The simulation cannot generally take into account all aspects of reality. A simulation model is thus simplified compared to reality and also applies only within a range defined by the assumed requirements. The defined range for state variables can generally be exceeded when solving differential equations. Typical cases include the calculation of the fill level of a container. In these cases, the state variables must be checked after solving the differential equation system and corrected if necessary.

SIMIT allows you to write a conditional equation for correcting state variables. You simply identify the correction value for the state by prefixing it with a # sign and write a correction equation as in the example below:

```
if (@Mass < 0.0) #Mass = 0.0;
```

This instruction causes the value for *Mass* to be set to 0.0 if the solver has calculated a value less than 0.0, otherwise the correction value for *Mass* is identical to the calculated value *@Mass*.

7.7.4.4 Accessing continuous state variables

As mentioned above, continuous state variables are also calculated cyclically. The values are thus only available at the times defined by the cycle. In a given time step you can only access the value in the previous calculation cycle and the newly-calculated or corrected value. As can be seen in the list below, you access the newly-calculated value for a state *Z* by prefixing it with an @ sign.

Table 7-12 Accessing continuous state variables

Notation	Access
Z	Value from the previous computation cycle
@Z	New value after integration of the state equation
#Z	Corrected value that is to be used in the next computation cycle instead of @Z

7.7.5 The equation-oriented approach

7.7.5.1 Overview

In the equation-oriented approach, the syntax in which you set out the behavior of your component type consists of relationships in the form of equations rather than instructions. These equations explicitly describe how a state variable or an output is calculated from other variables, for example as follows:

```
Output = Parameter * Input;
```

As a general rule:

- Every equation contains the equals sign.
- On the left of the equals sign is the variable that is being determined. On the right of the equals sign are the variables that are read (explicit form of an equation).

- At the end of every equation there is a semicolon.
- Any variable may occur only once on the left of the equals sign, which means a variable may only be determined once.

The following variables may appear on the left of the equals sign:

- Outputs
- States (for time-discrete states, only the new value; for continuous states, only the differential, which means the change in value) and
- Local variables.

The following variables may appear on the right of the equals sign:

- Inputs
- States (only the differential may appear here for continuous states)
- Parameters
- Local variables and
- Constants.

Note

The "equation-oriented approach" is obsolete and should not be used anymore, because it will not be supported in future versions.

7.7.5.2 Local variables

You can define local variables as follows in the behavior description for initialization or cyclic calculation:

```
Data_type Name[,Name];
```

Example:

```
binary b1, b2, b3;
```

The data types listed in the table below are permitted.

Table 7-13 **Data types for local variables**

Data type	Meaning	Value range	Default
binary	Binary values	True/False	False
analog	Floating point values	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$	0.0
integer	Integer values	–9.223.372.036.854.775.808 bis +9.223.372.036.854.775.807	0

The name of a local variable must contain only letters, digits and the underscore character, and must start with a letter.

Local variables are used to save interim results that will be needed again in the same processing step.

If you wish to access calculated values again in the next calculation step, always create time-discrete state variables rather than local variables.

7.7.5.3 Constants

The constants you can use will depend on the data type of the result variable. The constants for each data type are described in the table below.

Table 7-14 Data types for constants

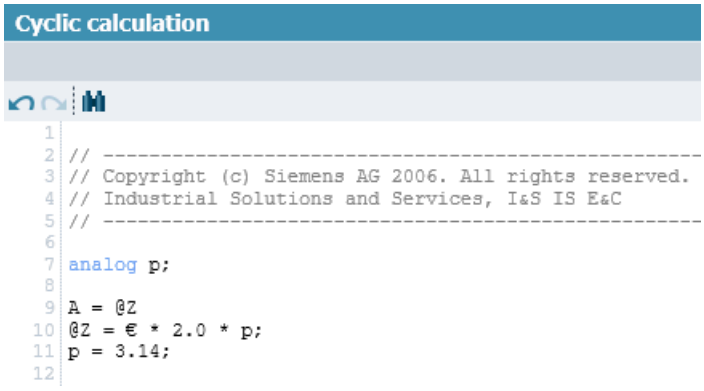
Data type	Constants
binary	"FALSE" or "TRUE"
analog	Decimal fraction with a dot as the decimal mark, for example "125.61" Exponential notation, for example "62.2e-4"
integer	Sequence of digits without thousands separator, for example "125985"

7.7.5.4 The calculation order

The behavior description for a component type both for initialization and for cyclic calculation consists of individual equations. With the description, you define only relationships and dependencies within the relationships. In particular, you do not define a calculation order. The order in which you write these equations is of no relevance to the calculation.

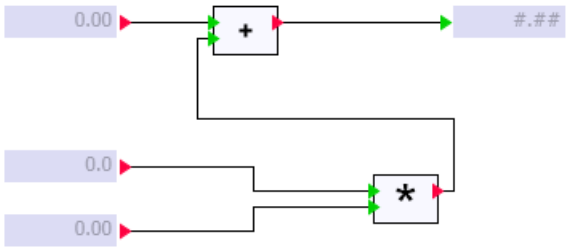
When you create an executable simulation, SIMIT analyzes all equations in a component instance and determines the order in which they are calculated based on the reciprocal dependencies. SIMIT always sorts the calculation order so that equations that determine a variable are calculated before equations that need this variable are calculated.

In the example below, the local variable *p* is first assigned the value 3.14, then the new state *Z* is calculated and then the value of the newly calculated state is assigned to the output.

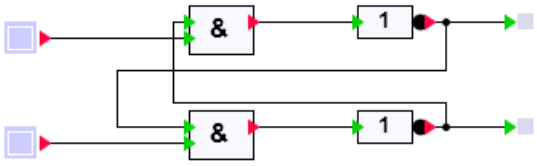


The calculation order is automatically obtained from the analysis of the dependencies and, in this case, is exactly the reverse of the order in the description.

However, SIMIT does not only analyze the equations within a single component instance, but rather across all the component instances with their reciprocal links in your SIMIT project and defines a suitable calculation order for the project. In the example below, multiplication is always executed before addition because the product is needed for the addition.



Situations may occur in which no unambiguous calculation order can be determined. In the example below, the input values of the two AND operations each depend on the output value of the other operation. This feedback means that one of the two AND operations is always calculated with a non-current input value, which means with a value originating from the previous calculation cycle.



In such cases, the order in which the equations are processed is undefined. SIMIT always tries to maximize the calculation order, that is, to sort as many equations as possible into one sequence. Thus, in the above example, SIMIT always processes the "Switch – AND – NOT – binary indicator" sequence in this order. It is, however, not defined whether the top or the bottom sequence should be processed first.

7.7.5.5 Operators

The variables on the right of the equals sign can be linked to one another with operators. The operators listed in the table below are permitted, depending on the data type.

Table 7-15 Permitted operators

Operation	Operator	Data type	Priority
Parentheses	(expression)	binary, integer, analog	highest
Function call	Function name (parameter list)	binary, integer, analog	
Change of sign	–	integer, analog	
Negation	NOT	binary	
Multiplication	*	integer, analog	
Division	/	integer, analog	
Modulo	%	integer	
Addition	+	integer, analog	
Subtraction	–	integer, analog	
Comparison	<, >, <=, >=	integer, analog	
Equality	=	binary, integer, analog	

Operation	Operator	Data type	Priority
Inequality	!=	binary, integer, analog	
Logical AND	AND	binary	
Logical exclusive OR	XOR	binary	
Logical OR	OR	binary	lowest

7.7.5.6 Conditional assignments

You can assign different values to a variable depending on one or more conditions, using the following syntax:

```
y = {Condition1: Expression1; Condition2: Expression2; etc.
            ELSE Expression0};
```

Example:

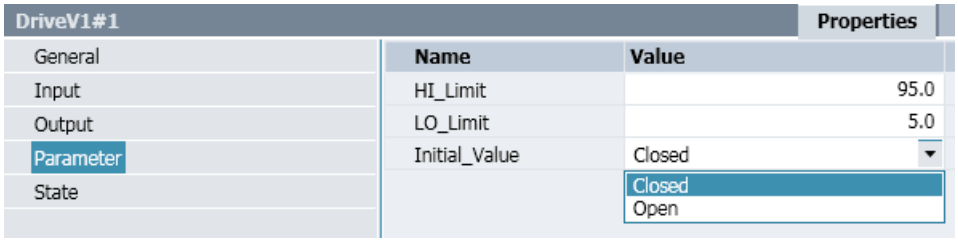
```
m = {p<1: m1; p<0: m2; ELSE m3};
```

The conditions are processed from left to right. Once a condition is fulfilled, the associated expression is evaluated and assigned to the variable on the left of the equals sign. If no condition is fulfilled, the expression after the keyword *ELSE* is evaluated.

You can specify as many condition/expression pairs as you wish. The "ELSE" expression must always be specified.

7.7.5.7 Enumeration types

Parameters can have enumeration types as the type. You can then select the desired entry from a list in the properties window of the chart editor to parameterize the component instance.



You can query such a parameter in the behavior description by referencing the name of the enumeration type followed by a dot and the corresponding list entry. For example:

```
A = {Initial_Value = ClosedOpen.Open: TRUE; ELSE FALSE};
```

7.7.5.8 Vectors

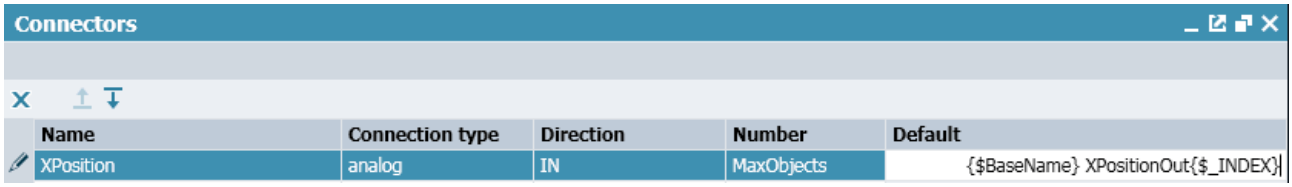
You can also define all inputs, outputs, parameters and states as vectors. You can then access the elements of the vector in the behavior description as follows:

- **Individual elements**
  To access an element of a vector, append the desired index in square brackets to the signal name.
  Note that the first element of the vector has the index *0*. The index must be a constant, non-negative integer value and must not exceed the number of elements that this vector has.
  Example:
  ```
  Output[2] = Input * 2.0;
  ```
- **Area**
  You can specify an equation for multiple elements of a vector by entering a range:
  ```
  Vector_name[Index1 TO Index2]
  ```
  SIMIT solves this vector equation so that the two examples below are identical:
  Example 1:
  ```
  Output[0 TO 2] = 1.0;
  ```
  Example 2:
  ```
  Output[0] = 1.0;
  Output[1] = 1.0;
  Output[2] = 1.0;
  ```
  When you want to refer to the index in a vector equation, you can specify an index variable:
  ```
  Vector_name[Index_variable: Index1 TO Index2]
  ```
  This index variable can then be used on the right side of the equation. It does not have to be declared as an additional local variable but its name must be a single character.
  Example:
  ```
  Output[i:0 TO 2] = Input[i + 1];
  ```
  In this case, the resolution into individual equations is as follows:
  ```
  Output[0] = Input[1];
  Output[1] = Input[2];
  Output[2] = Input[3];
  ```
- **Whole vector**
  If you want to extend the range to the whole vector, you can also write:
  ```
  Vector_name[ALL]
  ```
  or specify an index variable:
  ```
  Vector_name[Index_variable: ALL]
  ```
- **Vector of a complex variable**
  If a signal has a complex connection type, the index is specified in square brackets at the end of the complete name.
  Example:
  ```
  X1.Re[3] = Y1.Im[1] * Y2.Im[2];
  ```

_INDEX system variable

The *_INDEX* system variable is used for the variable definition of implicit vector connections.

You can use the *_INDEX* variable with the expression {$_INDEX} when specifying the signal name. When the component is instantiated this expression is then replaced in each element with the actual index, starting with 1.



If the following values have been set for a component as shown in this example,

- Parameter MaxObjects is set to 2 and
- Parameter BaseName is set to "LifterBase#1"

it results in the following default values for the input vector XPosition:

- XPosition1: LifterBase#1 XPositionOut1
- XPosition2: LifterBase#1 XPositionOut2

7.7.5.9 Function calls for mathematical standard functions

You can use all the mathematical standard functions anywhere in your behavior description where a scalar input variable is permitted.

Simply prefix the name of the mathematical standard function with an underscore and append a list of parameters in parentheses. The individual parameters are separated by commas. If a function has no parameters, it is called with an empty parameter list. The table below lists all the available mathematical standard functions.

Table 7-16 **List of mathematical standard functions**

Function	Return value	Description
_sqrt(x)	analog	$y = \sqrt{x};\ x \geq 0$
_abs(x)	analog/ integer	$y = \lvert x \rvert$
_exp(x)	analog	$y = e^x$
_pow(x, z)	analog	$y = x^z$
_log(x)	analog	Natural logarithm: $y = \ln(x);\ x > 0$
_log10(x)	analog	Common logarithm: $y = \lg(x);\ x > 0$
_ceil(x)	analog	Smallest integer greater than or equal to x
_floor(x)	analog	Largest integer less than or equal to x
_rand()	integer	Random value between 0 and 32767
_sin(x)	analog	$y = \sin(x)$; angle x in radians
_cos(x)	analog	$y = \cos(x)$; angle x in radians
_tan(x)	analog	$y = \tan(x)$; angle x in radians; $x \neq \pm(2n+1)\pi/2$
_asin(x)	analog	$y = \arcsin(x);\ -1 \leq x \leq 1$
_acos(x)	analog	$y = \arccos(x);\ -1 \leq x \leq 1$
_atan(x)	analog	$y = \arctan(x)$;

Function	Return value	Description
_atan2(y, x)	analog	$y = \begin{cases} \arctan(\frac{y}{x}) & x > 0 \\ \pi + \arctan(\frac{y}{x}) & y \geq 0, x < 0 \\ -\pi + \arctan(\frac{y}{x}) & y < 0, x < 0 \\ \frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ \text{Undefined} & y = 0, x = 0 \end{cases}$
_sinh(x)	analog	$y = \sinh(x)$; angle x in radians
_cosh(x)	analog	$y = \cosh(x)$; angle x in radians
_tanh(x)	analog	$y = \tanh(x)$; angle x in radians
_min(x, y)	analog/ integer	The smaller of the two values x or y
_max(x, y)	analog/ integer	The larger of the two values x or y
_trunc2byte(x)	byte	The integer part of x modulo 2^8
_trunc2int(x)	int	The integer part of x modulo 2^{16}
_trunc2long(x)	long	The integer part of x modulo 2^{32}
_round2byte(x)	byte	The rounded number x modulo 2^8
_round2int(x)	int	The rounded number x modulo 2^{16}
_round2long(x)	long	The rounded number x modulo 2^{32}
_characteristic(c[ALL], x)	analog	Characteristic is a special case (additional information is available in the section: The characteristic parameter type (Page 412))

7.7.5.10 User-defined functions

If you need your own operator you can define a user-defined function under the *Functions* sub-aspect in the behavior description. You can call user-defined functions in the *Initialization* and *Cyclic calculation* sub-aspects.

When you call a user-defined function, you can also set multiple variables if you have declared the function accordingly. When you call a user-defined function, always specify the list of calculated function values in parentheses.

Example:
```
(y1, y2) = UserFunction(x1, x2);
```

Even a single function value as a return value must be placed in parentheses.

You can only use user-defined functions with a call as shown above. Even if your user-defined function has only one return value, you cannot use this function like the mathematical standard functions as a scalar value in other relationships; the function must always be called as shown above.

The syntax used to describe functions differs significantly from the equation language. It follows the instruction-based approach. You can find additional information on this in the section: Functions (Page 394).

7.7.5.11 Accessing discrete state variables

For discrete state variables you define the initial value or the new value in each cycle with the aid of an explicit equation. The new value for a state *Z* is identified by prefixing it with an @ sign.

Table 7-17 Accessing discrete state variables

Notation	Access
Z	Value from the previous computation cycle
@Z	New value

7.7.6 Internal variables and constants

The system variables that you can use as constants are listed in the table below.

Table 7-18 System constants

Name	Data type	Description
_NAME	text	The name of the component instance
_TA	analog	Configured sampling time (=cycle time) in microseconds
_Time	analog	Current simulation time in milliseconds
_ta	analog	Configured sampling time (=cycle time) in seconds
_load	analog	– Reserved –
_scriptmode	binary	"True" while a script is running
_PI	analog	Pi (3.14159...)
_E	analog	Euler's constant (2.71828...)
_GRAVITY	analog	Gravitational constant (9.81)

In conveyor technology component types you may also use the system variables that are listed in the table below. These variables allow you to evaluate the dimensions of a component in its behavior description.

Table 7-19 System variables to determine component dimensions

System variable	Data type	Meaning
_WIDTH	analog	Width (unscaled) of the component in pixels
_HEIGHT	analog	Height (unscaled) of the component in pixels
_SCALEX	analog	Horizontal scaling of the component
_SCALEY	analog	Vertical scaling of the component
_TECHSCALE	analog	Scale of the chart on which the component is placed (in mm per pixel)

Table 7-20 System variables to determine the system time

System variable	Data type	Meaning
_t_msec	integer	Millisecond
_t_sec	integer	Second
_t_min	integer	Minute
_t_hour	integer	Hour
_t_day	integer	Day
_t_mon	integer	Month
_t_year	integer	Year

Note

If the system time or system date is changed, the system variables `_t_hour`, `_t_min`, `_t_sec` and `_t_msec` are not updated during the active simulation. If the simulation is restarted, these system variables are updated again.

7.7.7 The characteristic parameter type

Parameters can have the data type *characteristic*. It allows you to use one or more characteristics in a component. Note that you cannot create a vector of characteristics. The "number" of such a parameter is therefore always set to "1".

The use of such a parameter (P) that maps an input value (IN) onto an output value (OUT) differs according to the description approach used:

- **Instruction-oriented:**
  ```
  OUT = _characteristic(P, IN);
  ```
- **Equation-oriented:**
  ```
  OUT = _characteristic(P[ALL], IN);
  ```

7.8 Functional Mock-up Units (FMU)

7.8.1 Operating principle of the FMU integration

The Functional Mock-up Interface (FMI) defines a standardized interface that allows various simulation tools to be coupled together with little effort. The FMI interface contained in SIMIT allows precompiled models from external simulation tools, the so-called Functional Mock-up Units (FMU), to be integrated and executed in the SIMIT runtime environment.

FMUs that are compatible with the basic SIMIT simulation paradigm can thus be packaged in a SIMIT component type. One criterion of the simulation paradigm, for example, is the ability of dynamic, real-time capable execution in discrete time steps.

Supported FMU properties and compatibility (Page 414)

An export of SIMIT models to an FMU is not possible.

7.8.1.1 Conversion via component type editor / CTE

The conversion of an FMU to a SIMIT component type can be started in SIMIT CTE by selecting **New component** in the menu bar.

Like for a normal component, a component name and a target folder must be specified. In addition, the associated FMU is selected and the conversion operation is started with "Create".

The compatibility of the FMU is automatically checked. If the conversion is successful, a new SIMIT component type is created.

The following information is automatically applied from the *ModelDescription.XML* of the FMU:

- **Connectors** including description
- **Parameters** including description and unit
- **States** including description
- Supported functions are entered under **General > Change comments**

In addition, SIMIT-specific extensions are automatically added to the component:

- **Secondary parameters** under "Parameters > Secondary"
  Reading of FMU states including derivation (Page 417)
- **Initialization**: automatically generated behavior description
- **Cyclic calculation**: automatically generated behavior description
- **Basic symbol**

The embedded FMU is controlled and monitored with the help of the generated behavior description:

- Initializing the FMU
- Writing of component inputs to FMU
- Executing a simulation step in the FMU
- Reading the simulation results from the FMU and writing to component outputs
- Evaluation of the status of the FMU
- Optional: Reading the FMU states and writing to component states

Note

Manually adapting the automatically generated behavior description can lead to unpredictable errors. Moreover, the automatically generated connectors, parameters and states must not be adapted.

As soon as the conversion is complete and a SIMIT component type has been created, the embedded FMU can no longer be subsequently updated to a new version. If a new version of the original FMU exists, the conversion must be started again and a new component type created.

7.8.1.2 Conversion via SIMIT SP

It is possible to start the conversion directly from the SIMIT SP. As for normal component types, the conversion can be started via **Portal view > Simulation model > Create new component type**.

Conversion via component type editor / CTE (Page 413)

Note

A CTE license is needed for the conversion.

7.8.1.3 Placing and interconnecting the FMU components

In principle, the created FMU component can be placed and interconnected like any other SIMIT component using the chart editor. The following points are taken into consideration:

- **Assignment to a suitable time slice**:
  Under certain circumstances, an FMU may only take into consideration certain increments. Only the FMU supplier provides this information. The possibility exists for the FMU supplier to specify a preferred increment in the FMU description. If a value was specified for this, a corresponding comment is automatically entered in the **change comments** of the CTE component during the conversion. Only integrate the FMU in a time slice that conforms to this specified increment.
- **Multiple placement of a component type**:
  Some FMUs may only be instantiated once per component type. If such a component is placed more than once on a SIMIT chart, a corresponding message is generated.

7.8.2 Supported FMU properties and compatibility

The capabilities and properties of an FMU are checked before the conversion. In so doing, a distinction is made between properties that are necessary, supported and not supported. A detailed description of these properties can be found in the FMI specification (https://fmi-standard.org/downloads/).

7.8.2.1 Necessary properties

The necessary properties of an FMU must be satisfied. Otherwise, the conversion is canceled and a SIMIT component is not created.

The following properties are necessary:

Necessary properties	Description
FMU Version	Die FMU version must be 2.0 or 2.1. Older versions are not supported.
Co-Simulation	SIMIT supports only FMUs that are capable of being integrated as a co-simulation.
needsExecutionTool	SIMIT does not support FMUs that need a separate tool to execute the simulation. Execution of the simulation exclusively with the imported FMU file must be guaranteed.
Platform	SIMIT supports only win32 FMUs. The Binaries folder of the FMU must contain a win32 folder with the corresponding executable FMU libraries.
SourceCode FMUs	FMUs that exist exclusively in source code must be compiled for win32 before the import and stored again as an FMU file. SIMIT does not support compilation.
Enums	SIMIT does not support Enum-type ports.

7.8.2.2 Supported properties

In the case of supported properties, the values and data are processed and reacted to accordingly in the SIMIT environment.

The following properties are supported:

Supported properties	Description
canBeInstantiatedOnlyOnce	If this property is TRUE, an FMU can be used only once in a project. If the FMU is instantiated a second time when creating the project chart, an error message is sent.
stepSizeSpecified	An FMU can use this value to specify its preferred increment. A notice regarding the specified value is made in the change comments of the CTE component so that the user can set an appropriate time slice for the CTE component.
canNotUseMemoryManagementFunctions	An FMU uses its own functions for the memory allocation and deallocation.
Resource Folder	An FMU may need files for the internal processing in the fixed-named "resource" folder. The complete "resource" folder is transferred to the SIMIT component.

7.8.2.3 Unsupported properties

When unsupported properties occur, the FMU can still be converted. A notice is merely entered in the change comments field of the CTE component. The corresponding functions cannot be used in SIMIT, however.

Unsupported properties	Description
canHandleVariableCommunicationStepSize	In SIMIT, the time slices of each component are specified before a simulation. Time slices cannot be changed during a simulation.
canRunAsynchronously	An FMU is called synchronously in the component so that the answer of the FMU calculation is awaited before the evaluation of the next component is called.
canGetAndSetFMUState and canSerializeFMUState	Saving and restoring FMU states is not supported.
canInterpolateInputs	SIMIT components cannot provide derivation information of continuous outputs.
providesDirectionalDerivatives	SIMIT has no increment control and cannot utilize Jacobi information.
ModelStructure	Information regarding the dependence of the outputs and state derivations on inputs and states is not further used.

7.8.3 Automatic compatibility adaptations

During the conversion, various compatibility adaptations may be made automatically to allow execution of the FMU in SIMIT.

Adaptation of names

Names of connectors, parameters, states and enumeration types are adapted during conversion to a SIMIT component because SIMIT allows only a limited character set.

The following rules are applied during the conversion:

- All characters that are not letters or digits from the ASCII character set are replaced with an underscore "_".
- If the name starts with a number or an underscore, "FMU_" is inserted as a prefix at the start of the name.
- The name must be unique. For this reason, names that occur multiple times after the above conversion are made unique by adding a suffix "_{i}". {i} represents an integer that counts up from 1.

Tuneable parameters

Parameters that can be changed at runtime are designated as **tuneable parameters** in FMUs and are created in SIMIT as **parameters** with the **Modifiable online** property.

Calculated parameters

FMUs can contain **calculated parameters**. The **calculated parameters** are listed in SIMIT as **states** but not as parameters. That is because they are calculated within the FMU and cannot be changed by the user.

Enumerations

FMU-specific enumerations for parameters are applied in SIMIT, and a corresponding SIMIT enumeration type is created.

SIMIT does not support enumeration types for states. FMU enumerations for states or calculated parameters are thus not exchanged with SIMIT. They are correctly handled within the FMU, however.

The elements of an FMU Enum can be assigned any (unique) integer. Even if the elements of an Enum in SIMIT are always consecutive integer values (starting with 0), this property is supported by a corresponding transformation.

7.8.4 Extended functions and diagnostics

7.8.4.1 Reading of FMU states including derivation

During conversion of an FMU, **secondary parameters** are automatically added to a component that enable the reading of FMU states including their derivations.

The following are the secondary parameters.

- getFMUStates = TRUE: States of an FMU are read cyclically. The corresponding state tags of the SIMIT component are written.
- getFMUStateDerivatives = TRUE: Derivations of states of an FMU are read cyclically. The corresponding state tags of the SIMIT component are written.

For performance reasons, these functions are disabled by default.

7.8.4.2 Diagnostic capabilities of an FMU under error conditions

From the SIMIT perspective, imported FMUs are a black box and are merely called.

If errors occur in the FMU process, they are relatively difficult to evaluate in SIMIT or can be evaluated only to a limited extent. Only information that the FMU itself provides externally can be used within SIMIT.

An important aid for this is the automatically generated output "*Fmi2Status*" that every FMU component receives in SIMIT. General status and states of an FMU can be evaluated with this.

Fmi2Status	Description	Reaction of SIMIT
0	Fmi2Ok	Continue with next simulation step
1	Fmi2Discard	Current simulation step of the FMU is discarded and old values are output. FMU continues to be called cyclically
2	Fmi2Warning	Current simulation results of the FMU are output. FMU continues to be called cyclically

Fmi2Status	Description	Reaction of SIMIT
3	Fmi2Error	The FMU instance is no longer called cyclically. The other FMU instances are still called cyclically. When simulation is stopped: clean-up and controlled ending of the FMU instance.
4	Fmi2Fatal	All instances of the FMU instance are no longer called cyclically. No function calls for clean-up and controlled ending are issued either.

As soon as *Fmi2Status>=1*, a corresponding error message is issued under **Monitoring > Messages**.

With the secondary parameter "*lastValidOutputValuesOnError*", the behavior of the outputs under error conditions can also be set (*Fmi2Status>=3*).

Fmi2Status	Reaction under error conditions
0	0 and NaN are output
1	The last valid values are output.

If the secondary parameter *"EnableDebugTraces*" is set to "TRUE", extended diagnostics information is displayed in the SIMIT trace view.

Libraries

8

8.1 Basic library

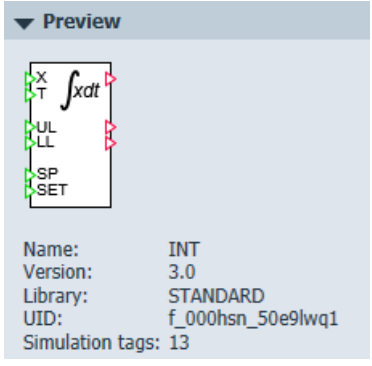
8.1.1 General

8.1.1.1 Introduction

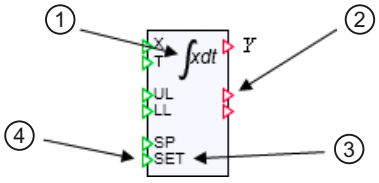
The basic library of SIMIT contains elementary functions for creating simulations, i.e. for modelling plant and machine behavior. These functions are provided in the form of component types and controls. This following sections describe in detail the individual component types and controls contained in the SIMIT basic library.

8.1.1.2 Component symbols

Component types are instantiated as components in order to create a simulation. To do this, select the required component type with the mouse and drag it into a chart. Each instance of a component is represented by a type-specific symbol in a chart. The symbol for a component type, its name, its version, the library to which it belongs, its UID and the number of simulation tags are displayed in the preview in the figure below. Click on the component type in the library to select it.

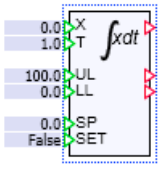


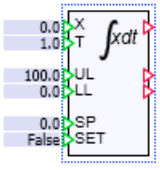
Every symbol has connectors with names and a label or graphic that clearly shows the function of the component on charts (see the figure below). The symbols are designed so that the function of both the component and the connectors can be understood intuitively.

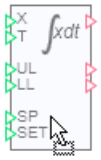


① Labeling / graphics
② Output connector
③ Connector name
④ Input connector

Components are represented by the type-specific symbol in charts. Click on the relevant symbol to select a component in a chart. A blue frame then appears around the symbol for the selected component. Simply hold down the mouse button to drag the symbol and to move it within the chart.

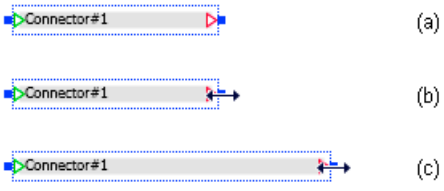




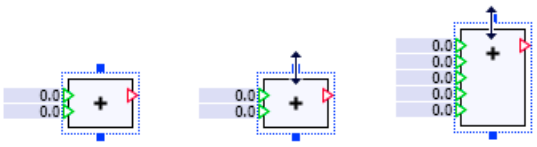


Some components have handles on the selection frame. These handles are used to change the size of the symbol.

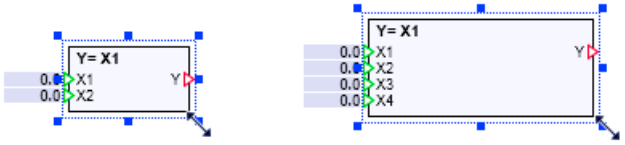
Components such as connectors have handles on the left and right of the selection frame as shown in the figure below under (a). When you roll the cursor over the handles, its appearance changes as shown in the figure below under (b). Hold down the left mouse button to move the handles and thus change the width of the symbol as shown in the figure below under (c).



For components such as *ADD*, the selection frame has handles at the top and bottom. These handles are used to adjust the height of the symbol to suit the number of inputs. Simply click on the top or bottom gripper, hold down the left mouse button and drag it up or down.

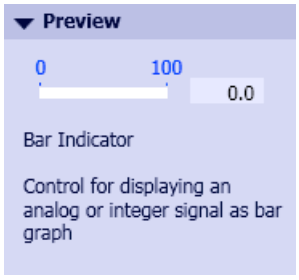


The formula components have handles on all sides and at every corner of the selection frame. These allow you to adjust both the width and the number of inputs. The handles at the corners of the symbol allow you to make these two settings at the same time.

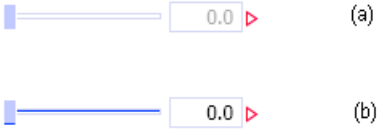


8.1.1.3 Symbols for the controls

When you create a simulation, controls are handled in the same way as component types, which means they are positioned in a chart with their symbol. Controls that you selected from the library appear in the preview with their symbol, their designation and a brief description of their function.



When the simulation starts, controls act as active elements and they are represented accordingly as active controls by their symbols as seen in figure below under (b). If there is no active simulation, the symbols represent passive controls and are displayed as such as seen in figure below under (a).



Controls, like components, are represented by the type-specific symbol in charts. Simply click on a symbol to select it. The symbol of the selected control then appears with a blue selection frame; simply hold down the left mouse button to drag the symbol and move it within the chart. Use the handles on the selection frame to change the size of the symbol.

8.1.1.4 Structure of the simulation model

A simulation model in SIMIT is created by placing components from the available libraries (predefined or self-defined) onto charts. The connectors of the individual components are then connected to each other. Only connectors of the same type can be connected.

Connectors to the flow network are represented by a black unfilled circle on the components. Connectors to the flow network that are connected are represented by a gray filled circle. If multiple components are connected via one connector, an internal node is created automatically. This is represented by a black dot in the connecting line. All flow network connectors must be connected.

Other connectors are represented using triangles.

- A red unfilled triangle signifies an unconnected output.
- A red filled triangle signifies a connected output.
- A green unfilled triangle signifies an unconnected input.
- A green filled triangle signifies a connected input.

Inputs and outputs do not have to be connected.

8.1.1.5 Component connectors

Component connectors from the Basic Library are inputs or outputs. Inputs (green triangles) are arranged on the left and outputs (red triangles) appear on the right of the symbol. To visually emphasize the direction in which the connectors work, the input triangles point into the symbol while the output triangles point out of the symbol.

Inputs and outputs that belong together from a functional point of view are arranged opposite one another in the symbol for a component if possible. In the above example of the integrator, these are input *X* and integrator output *Y*, for example. Inputs and outputs that belong together from a functional point of view are also grouped together and are separated from other groups by spaces. This means that the functional interactions created by interconnecting components can be more easily identified in charts. In the example from the section: Component symbols (Page 419), the following three groups are formed:

- Inputs *X* and *T* for calculating the integral value at output *Y*,
- the limits *UL* and *LL* with their binary feedback and
- set point *SP* and set command *SET* for setting the integrator output.

The function described by the integral

$$Y = \frac{1}{T} \int X \, dt$$

can thus easily be assigned to connectors *X*, *T* and *Y*. Connectors are only given names in the symbol if the function of that connector is not obvious.

All inputs and outputs are either binary (logical), analog or integer inputs or outputs. Complex connector types are not used in the components of the Basic Library. One exception is the *PROFIdrive* type connector that is used to connect the header component and the device-specific component in the PROFIDRIVE library. The values of the binary inputs or outputs are designated by zero and one or, alternatively, by False and True.

8.1.1.6 Connectors for controls

Controls for entering signals have only one output as their connector in form of a red triangle on the right side of the symbol (see the table below). A green triangle on the left side of the symbol identifies the input to the display controls. The signal splitter control has only one connector which is always invisible.



Input control slider



Output-control bar graph display

As with components, the connectors for controls are either binary (logical), analog or integer inputs or outputs.

8.1.1.7 Connecting I/O

The connectors for controls and components can be connected to one another if the following rules are observed:

1. Only inputs may be connected to outputs.
2. An input can only be connected to one output, while an output can be connected to multiple inputs.
3. Connectors to be connected must be of the same type.

Connectors can be connected in different ways:

- Connecting lines
- Placing connectors on top of each other
- Through implicit connections

In the first two cases, the connection is made graphically in the work area of the chart editor. The chart editor is designed so that the rules for connecting connectors defined above are automatically followed.

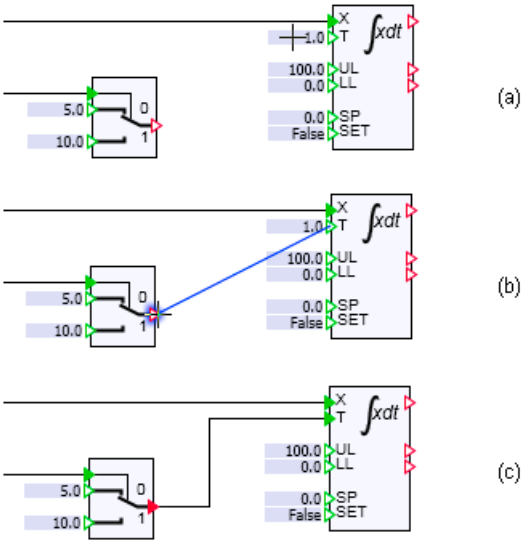
Implicit connections are made by changing the settings in the properties of the inputs or outputs of components to be connected.

Connecting with connecting lines

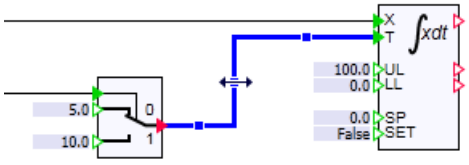
If you want to create a connection between the output of the *Selection* component and the input *T* of the integrator, for example, move the cursor over one of the two connectors. When the cursor changes to a cross, as seen in the figure below under (a), you can click or hold down the mouse button to create the connection. If you then move the cursor, a blue rubber band indicates the connection between the connector and the cursor.

Now move the cursor over the connector to be connected. When the connector to be connected is visually highlighted, as seen in the figure below under (b), the connection can be completed. If you held down the mouse button to open the connection, release the button to close the connection. If you opened the connection with a mouse click, click on the highlighted connector

to close the connection. When the connection is closed, the rubber band is automatically replaced with a connecting line with right angles and the triangles of the connected connectors are filled in with color, as shown in the figure below under (c).



To delete connecting lines, first click on the connecting line to be deleted. The connecting line then changes to a thick blue line as in the figure below and can be deleted by going to "Edit > Cut" in the menu bar or with the "Del" delete key.

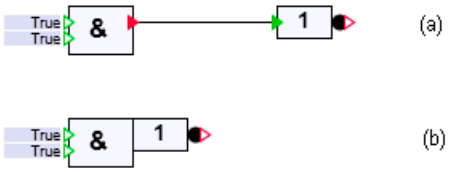


If a connection line selected, this connection can be manually edited.

Connecting by superimposing connectors

Connecting by superimposing connectors is done by positioning two components and/or controls to be connected on the chart so that the input of one component lies directly above the output of the other component to be connected.

The figure below compares this type of connection (b) with the connecting line method (a). The two connectors that are connected by superimposition become invisible.



Note

If the connectors do not become invisible when they are overlapping, the connectors are not of the same type and thus cannot be connected.

Implicit connections

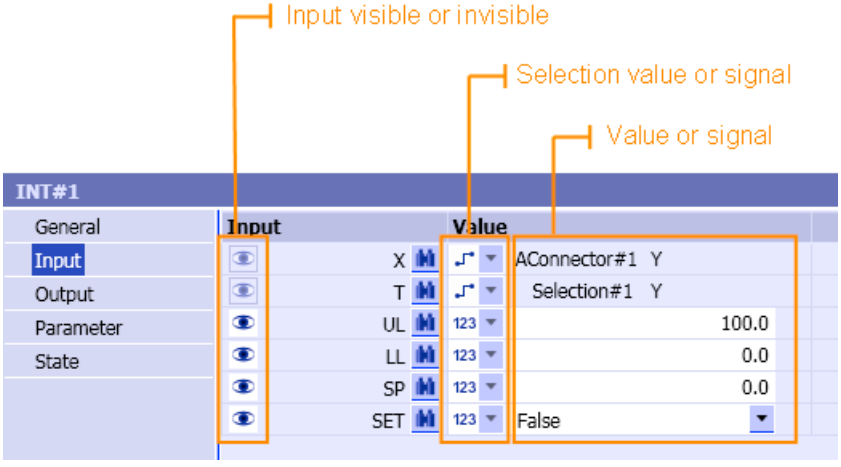
Implicit connections are made by changing the settings in the properties of the inputs and outputs to be connected of components and/or controls. To do this, open the property view of the component (see figure below) and follow these steps:

1. Make the input or output to be connected invisible ( ).
   Click the or symbol to toggle between input visible and input invisible.

   Note

   An invisible input or output is not displayed on the symbol for the component and thus cannot be connected to another output or input using connecting lines.

2. Set the selection field for value/signal specification to signal specification ( ).
3. Enter the signal to be connected with component name (source) and connector name.



The procedure is the same for a control. Because a control does not allow you to select a value/signal, the second step is omitted and the output to be connected may be entered directly after toggling to invisible.

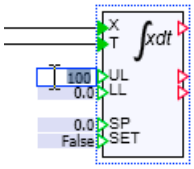
8.1.1.8 Setting inputs

Unconnected inputs may be preassigned with values. You can enter the value in the

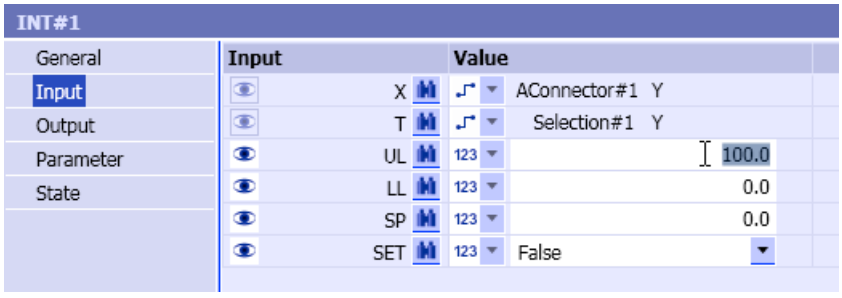
- connector box for the input on the chart or in the
- property view for the component

.

Double-click in the connector box to open the field for entering the value (see figure below). To complete your input, either press Return or click in the chart outside the connector box.



To enter a value in the property view, navigate to the relevant input and click in the input field to open it (see figure below). To complete your input, either press Return or click in the property view outside the input field.



Input of True and 1 or False and 0 are equivalent for binary parameters. Binary values are always displayed as True or False.

You can set inputs in the ways described above even when the simulation has started. In this case, however, the input will only take effect for the duration of the started simulation, which means modified input values are reset to their original values when the simulation ends.

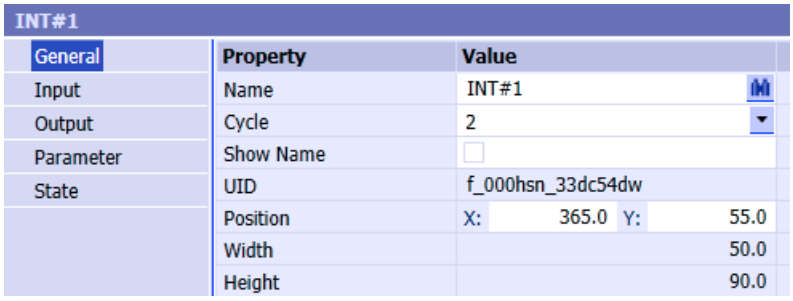
8.1.1.9 Properties of components

The properties of components can be accessed in the properties window. To display the properties of a component in the property view, right-click or left-click on the component. If a simulation is running, you can only display the properties by right-clicking. The properties of a component are divided into:

- General properties
- Properties of the inputs
- Properties of the outputs
- Parameters
- States

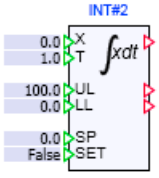
General properties

General properties of components are the name and the time slice of the component, the unique identifier (UID) of the component type, the position of the component and the width and height of the symbol.



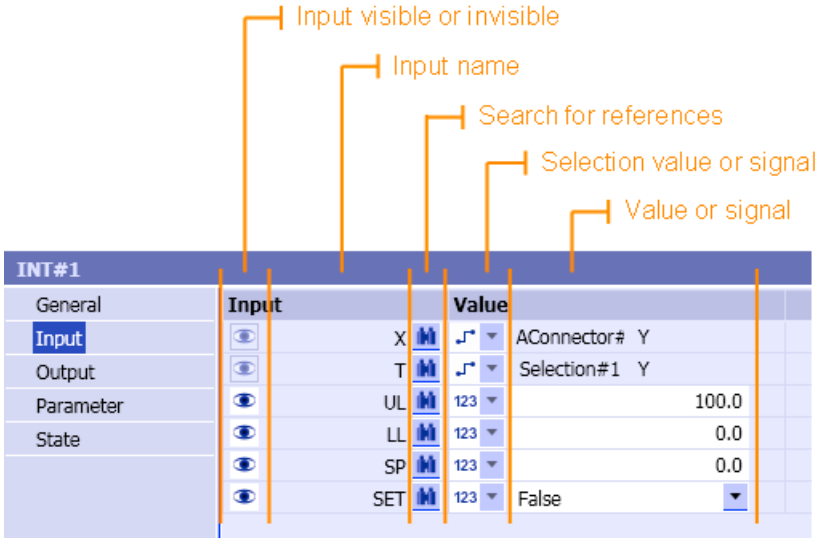
The name must be unique for all components and controls used in the project, which means the project must not contain multiple components and/or controls with the same name. When you drag a component from the library onto a chart, it is automatically assigned a name. This name is made up of the designation of the component type and a number for the component type that is unique across the entire project.

Check the *Show Name* check box to display the name of the component on the chart.



Properties of the inputs

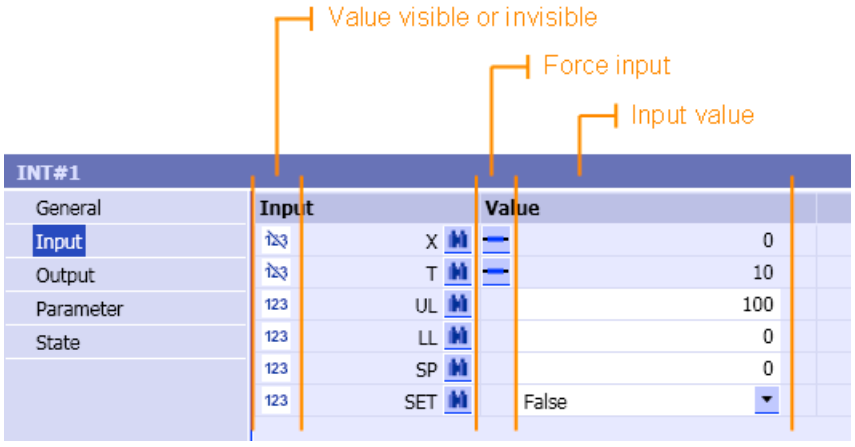
Inputs may be visible or invisible and the signal linked to an input or the input value may be set.



Each input has the following properties:

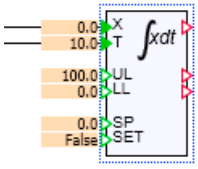
- **Visibility**
  In the first column, you can toggle between Input visible ( ) and Input invisible ( ). If the input is connected with a connecting line, you cannot toggle between them. In the figure above, for example, the two first inputs *X* and *T* are connected; they can therefore not be set to invisible.
- **Name**
  The name of the input is displayed right-justified in the second column.
- **References**
  The third column is used to search for references, which means for objects that use this input.
- **Selection value or signal**
  The fourth column is used to toggle between value ( ) or signal ( ) as the selection for the input. This selection is not active if the input is connected with a connecting line.
- **Value or signal**
  In the fifth column, the input value is set if value was selected. If signal was selected, the output connected with the connecting line is displayed or the output to be connected implicitly may be set.

The representation of the properties changes when the simulation is running.



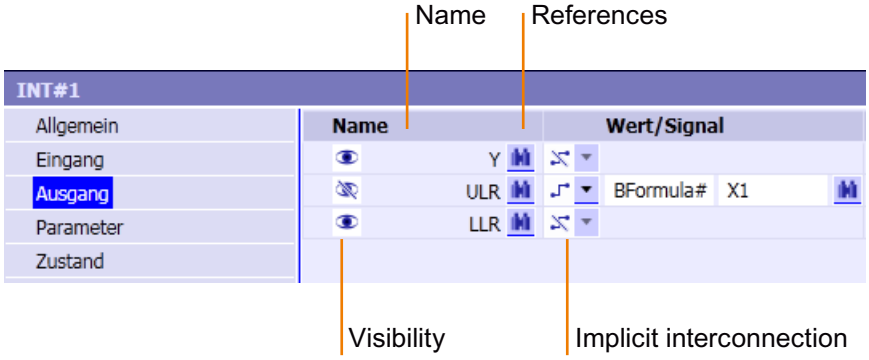
In this case input values are always displayed instead of signals, and it is possible to set each input value.

- **Display On/Off**
  In the first column, the value at the input of the component can be shown (123) or hidden (123). When the simulation is started, the display is switched on for non-connected inputs and switched off for connected inputs. The display cannot be switched on for invisible inputs. The figure below shows a component in which the display is switched on for all input values, which means even for values at connected inputs:



- **Forcing input**
  In the fourth column you can switch forcing on (T) or off (—) for every connected input.
- **Input value**
  The fifth column is used to display or set the input value.

Properties of the outputs



Each output has the following properties:

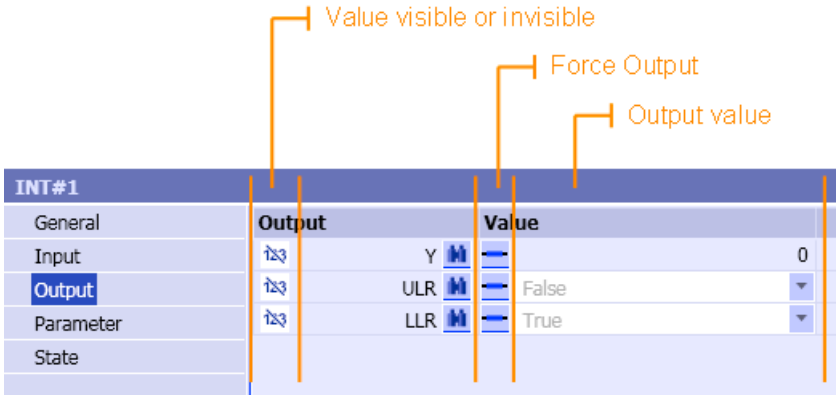
- **Visibility**
  In the first column, you can toggle between visibility (👁) and invisibility (👁) of the output. If the output is connected with a connecting line or by overlapping connectors, you cannot toggle between them.
- **Name**
  The name of the output is displayed right-justified in the second column.

- **References**
  The third column is used to search for references, which means for objects that use this output.
- **Implicit interconnection**
  In the fourth column, you can set whether this output is to be implicitly connected.

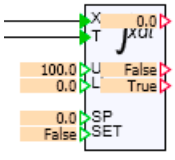
	Output is implicitly connected
	Output is not implicitly connected

If implicit connection is selected, an input field opens. Enter here the signal source to be interconnected and name.

The representation of the properties changes when the simulation is running.



- **Display On/Off**
  In the first column, the value at the output of the component on the chart can be shown (123) or hidden (123). When the simulation starts, the display is switched off for all outputs. The display cannot be switched on for invisible outputs. The figure below shows a component in which the display is switched on for all output values.



- **Forcing output**
  In the fourth column, you can switch forcing on ( ) or off ( ) for every connected output.
- **Output value**
  The fifth column is used to display or set the output value.

Displaying the input and output values of components

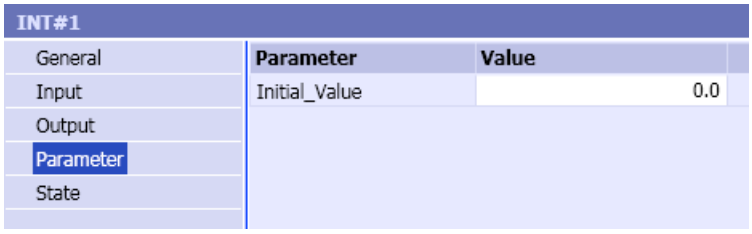
When the simulation is running, the display of individual current input and output values of components can be switched on in the property view of the component by means of the button. To switch on the display of all inputs and outputs of components on a chart with a single mouse click, you can use the same command in the chart toolbar.



If you select one or more components before executing the command, the switch applies only to the selected components. If no components are selected, the switch applies to all components on the chart.

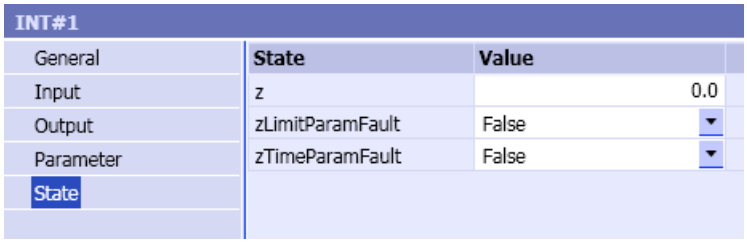
Parameters

Each parameter is displayed with its name and value in the property view:



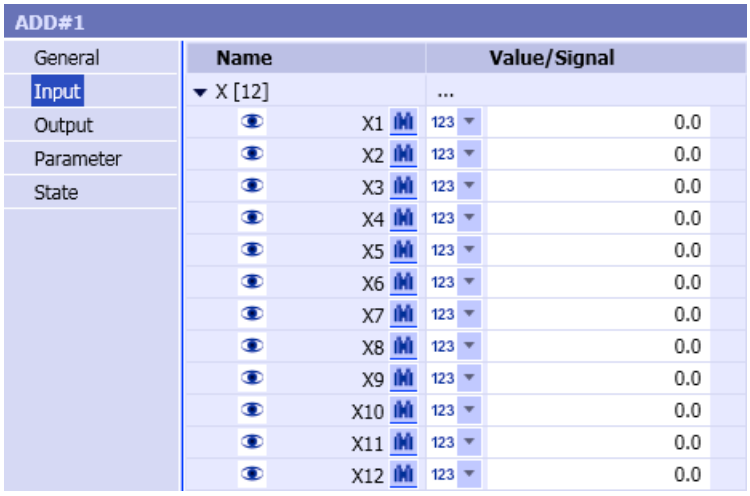
States

States are displayed with their name and initial values in the property view. When the simulation is started, the current values are displayed for each state.

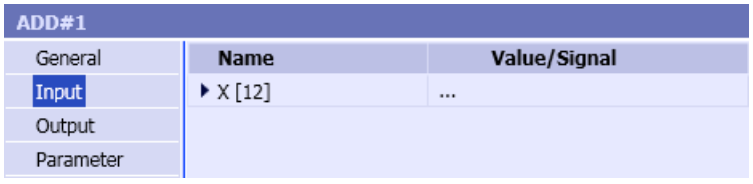


Displaying vectors in the properties window

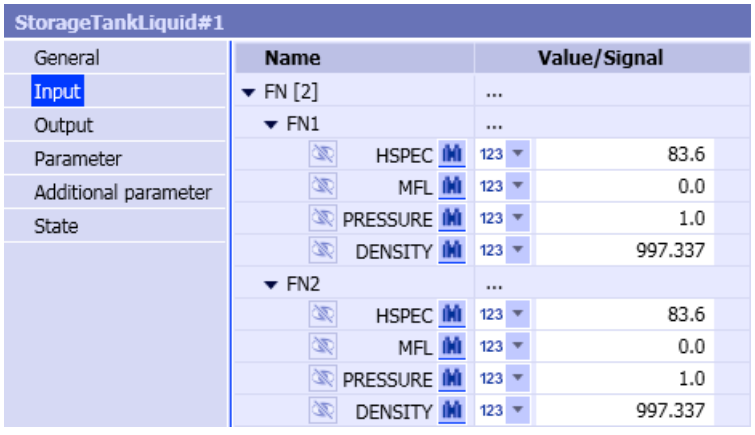
Vectors of inputs, outputs, parameters and states are displayed in the property view grouped and in a numerically correct sequence, as shown by way of example in the figure below.



The vector elements can be expanded and collapsed. The figure below shows a vector with collapsed elements.



For connectors with complex connection types an additional expandable and collapsible level has been added.



8.1.1.10 Component error messages

The components are implemented so that critical or nonsensical parameters or input values do not cause unstable component behavior. The component outputs an error message if impermissible values are specified or if input signals are not within the designated range.

In addition, outputs of the component can be set to a defined value in the event of an error to avoid unstable output values. This set value remains in effect until the error condition is eliminated.

All error messages from components in the basic library are assigned to the *ERROR* message category.

Error messages are generated as so-called incoming and outgoing messages. The incoming message is generated when the error occurs; the outgoing message is generated when the error condition has been resolved. Both messages have the same message text; the difference is that the text of outgoing messages is placed in parentheses.

Note

If you use the *ERROR* message category in your own messages which you generate using the *Message* component from the Trend and Messaging Editor, for example, you will be unable to distinguish between messages from components of the basic library and your own messages based on the message category.

8.1.1.11 Properties of controls

The properties of controls can be accessed in the property view. To display the properties of a component in the property view, right-click or left-click on the component. If a simulation is running, you can only display the properties by right-clicking. Each control has

- General properties and
- Properties for the connector.

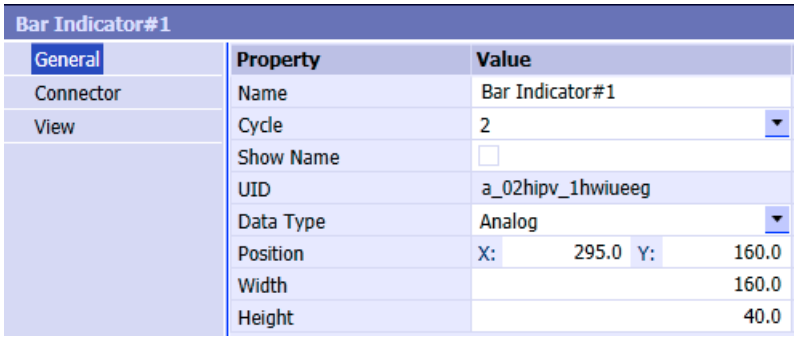
Controls whose representation can be changed also have

- Properties for the view.

You can find additional information on this in section: Controls (Page 584).

General properties

General properties of controls are the name, the time slice, the unique identifier (UID), the position as well as the width and height of the control.



The name must be unique for all components and controls used in the project, which means the project must not contain multiple controls and/or components with the same name. When you drag a control from the library onto a chart, it is automatically assigned a name. This name is made up of the designation of the control and a number for the control that is unique across the entire project.

Check the *Show Name* check box to display the name of the control on the chart.

Controls may also have other specific, general properties. You can find additional information on this in the section: Controls (Page 584).

Properties of connectors

Connectors may be visible or invisible. The figure below shows the property view with the connector of a control by way of example.



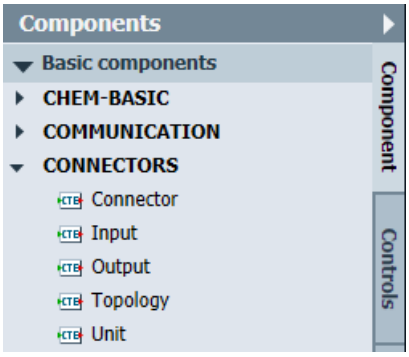
Each connector has the following properties:

- **Visibility**
  In the first column, you can toggle between visibility ( ) and invisibility ( ) of the connector. If the connector is connected with a connecting line or by superimposing connectors, you cannot toggle between them.
- **Name**
  The name of the connector is displayed right-justified in the second column.
- **Signal**
  In the third column, the connected signal is displayed or can be set for invisible connectors.

8.1.2 Connectors

The *CONNECTORS* directory of the Basic Library contains connectors:

- A global *connector*,
- The I/O connectors, *Input* and *Output*
- The special connector, *Unit*
- The *Topology* connector is only used with special SIMIT modules or libraries and is therefore not described here.



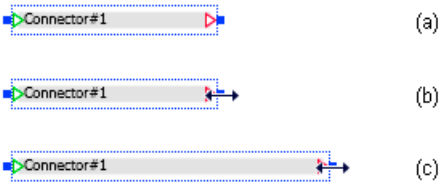
The connectors of the basic library have the following shared characteristics:

- The connectors of the global connector and of the I/O connectors have no type. This means the connectors assume the connection type of the connector of the connected component or control.

Note

A type check, which means a check to identify whether the connections connected via connectors are of the same type, is carried out automatically before the simulation is started. If connectors are connected with connectors of an incorrect type, a corresponding message is output by the consistency check and the start of the simulation is cancelled.

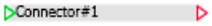
- The width of the connector symbol on a chart can be adjusted to match the length of the connector name. To set the width of a connector, click on the symbol. A blue frame with handles appears on the right and left of the frame (a). If you move the cursor by means of a gripper, the shape of the cursor changes (b). Then press and hold down the left mouse button to change the width by moving the mouse pointer to the to the left or right (c).
- The connector name is displayed in the connector symbol.



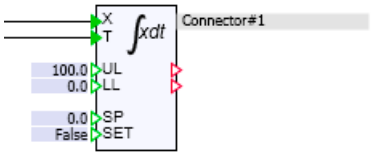
8.1.2.1 Global connector

The global connector *Connector* is used to connect components and/or controls across the boundaries of individual charts.

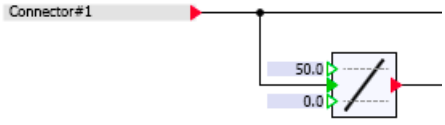
The global connector may be used as either an output connector or an input connector. Its symbol is illustrated in the figure below.



If the global connector is connected to the output of a component or control, the connector on the right side of the connector disappears: the connector is now used as an output connector.



If the global connector is connected to the input of a component or control, the connector on the left side of the connector disappears: the connector is now used as an input connector.

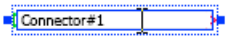


For a connection between an output connector and one or more input connectors, all of these connectors receive the same name: the connector name.

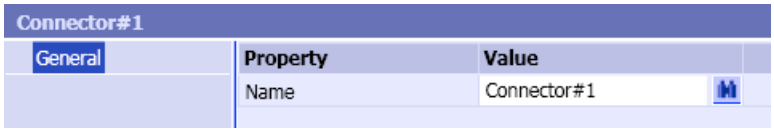
Note

Global connector name must be unique in the entire simulation project.

If you drag the global connector from the library onto a chart, the connector is assigned a name that consists of the term *Connector* and a unique sequence number. A connector name can be entered directly in the symbol. Double-click on the connector to open the input box, then press Return or click outside of the input box to confirm your input.



The connector name can also be entered in the property view. The connector name is the only property of the global connector.



8.1.2.2 I/O connectors

I/O connectors establish the link to signals in the SIMIT couplings. An I/O connector can establish a link to signals from any SIMIT coupling.

I/O connectors exist in the form of *Input* and *Output* connectors as shown below.



Input connector *Input*



Output connector *Output*

Connections between coupling signals and I/O connectors can be

- a connection from one or more *Output* connectors to a single output signal of a coupling, or
- a connection from a coupling input signal to an input connector *Input*.

The connection is established by setting the name of the coupling and the name of the coupling signal in the property view of the I/O connector.



If you do not want the name of the coupling to be displayed on the chart, you may deselect the option *Display coupling name*.

8.1.2.3 Topological connector

Topological connections can be realized beyond the respective chart borders using topological connectors. This connector is present in the CONTEC and in the FLOWNET library. The basic method of operation is the same.

You can find information on the topological connector in the CONTEC library in the section: Topological connector in the CONTEC library (Page 886).

You can find information on the topological connector in the FLOWNET library in the section: Topological connector in the CHEM-BASIC and FLOWNET libraries (Page 616).

8.1.2.4 Unit connector

The *Unit* connector is a special type of connector that can only be used in conjunction with "SIMATIC components" or components with integrated data record communication. In combination with these components, the unit connector establishes relationships between components and modules of the following couplings:

- SIMIT Unit
- Virtual controller
- PLCSIM

- PLCSIM Advanced
- PRODAVE

To configure a *Unit* connector using the automated import mechanism, the language-dependent properties "Coupling" and "Addressing" must be addressed with the terms "Gateway" and "Info".

See also

Access to a data record or memory area (Page 216)

Components for SIMATIC (Page 570)

8.1.3 Standard components

8.1.3.1 Differentiating standard components

The standard component types in the Basic Library form the Standard Library. They are contained in the *STANDARD* directory. The component types are divided into component types according to function with

- Analog functions,
- Binary functions,
- Integer functions,
- Conversion functions,
- Mathematical functions and
- Various auxiliary functions.

8.1.3.2 Analog functions

General information about the analog functions

All component types with analog functions are contained in the *AnalogBasic* and *AnalogExtended* directories of the Standard Library. *AnalogBasic* contains the basic analog functions, while *AnalogExtended* holds the extended analog functions.

Basic analog functions

Introduction

The four basic analog functions of Addition, Subtraction, Multiplication and Division, which means the four basic arithmetic operations, are stored as component types in the *AnalogBasic* directory of the Standard Library.

ADD – Addition

Symbol



Function

The *ADD* component type maps the sum of the analog values at the *n* inputs x_1 to x_n onto the output *y*:

$$y = \sum_{i=1}^{n} x_i = x_1 + x_2 + \ldots + x_n$$

The number of inputs *n* is variable and can be set to any value between 2 and 32. All inputs are set to zero by default.

SUB - Subtraction

Symbol



Function

The *SUB* component type maps the difference between the analog values at the inputs x_1 and x_2 onto the output *y* according to

$y = x_1 - x_2$

All inputs are set to zero by default.

MUL - Multiplication

Symbol



Function

The *MUL* component type maps the product of the analog values at the *n* inputs x_1 to x_n onto the output *y*:

$$y = \prod_{i=1}^{n} x_i = x_1 x_2 \ \ldots \ x_n$$

The number of inputs *n* is variable and can be set to any value between 2 and 32. All inputs are set to one by default.

DIV – Division

Symbol



Function

The *DIV* component type maps the quotients of the analog values at the inputs x_1 and x_2 onto the output *y* according to

$y = x_1 / x_2$

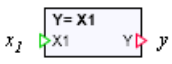
Input x_1 is set to zero by default, while the divisor input x_2 is set to one by default.

The value of the divisor x_2 must not be zero. If the divisor becomes zero during the simulation, the error message "*DIV: division by zero*" (message category *ERROR*) is generated and output *y* is set to zero.

Extended analog functions

AFormula – analog formula component

Symbol



Function

The component type *AFormula* allows the use of explicit algebraic functions. This function *f* calculates a value in relation to the *n* input values x_i. The function value is assigned to the output *y*:

$y = f(x_1, \ldots, x_n)$

The number *n* of inputs can be varied between 1 and 32. Only the inputs that are available according to the number currently set may be used in the formula.

The following operators are used in expressions formula:

Table 8-1 Operators in formulas for the AFormula component

Operator	Function
+	Addition
–	Subtraction
/	Division
*	Multiplication
(	Opening parenthesis
)	Closing parenthesis
Numeric constant	Floating point numbers, also in exponential notation
Function calls	Standard mathematical functions

Note

When entering floating point numbers, use a decimal point (and not a comma).

Note

There is also no check in the formula component to determine whether the arguments of the formula have valid values. If division by zero occurs during the simulation in the formula calculation, output *y* has the value **Inf**. If an argument of a formula is undefined during the simulation, as in the case of a division of zero by zero, output *y* has the value **NaN** (not a number).

These irregular output values will then be propagated in the model in all values that depend on this output. Your simulation will thus enter an undefined state. To avoid this situation, make sure that the inputs of the formula component can only assume values that ensure the validity of the arguments in the formula.

There is no check in the formula component to determine whether all the set inputs are used in the specified formula.

The mathematical functions listed in the table below are available as standard mathematical functions.

Table 8-2 Mathematical functions in formulas for the AFormula component

Formula	Function
sqrt(x)	$y = \sqrt{x}$; $x \geq 0$
fabs(x)	$y = \lvert x \rvert$
exp(x)	$y = e^x$
pow(x, z)	$y = X^z$;
log(x)	Natural logarithm: $y = \ln(x)$; $x > 0$
log10(x)	Common logarithm: $y = \lg(x)$; $x > 0$
ceil(x)	Smallest integer greater than or equal to *x*
floor(x)	Largest integer less than or equal to *x*

Formula	Function
rand()	Integral random value *y*, $0 \le y \le 32767$
sin(x)	$y = \sin(x)$; angle *x* in radians
cos(x)	$y = \cos(x)$; angle *x* in radians
tan(x)	$y = \tan(x)$; angle *x* in radians; $x \neq \pm(2n+1)\pi/2$
asin(x)	$y = \arcsin(x)$; $-\pi/2 \le y \le \pi/2$
acos(x)	$y = \arccos(x)$; $0 \le y \le \pi$
atan(x)	$y = \arctan(x)$; $-\pi/2 \le y \le \pi/2$
atan2(z, x)	$y = \arctan(x / z)$; $-\pi \le z \le \pi$
sinh(x)	$y = \sinh(x)$; angle *x* in radians
cosh(x)	$y = \cosh(x)$; angle *x* in radians
tanh(x)	$y = \tanh(x)$; angle *x* in radians

Note

You can use the function *y*= rand() to generate random numbers *y* within a specified range YMIN ≤ *y* ≤ YMAX using the formula YMIN + rand() * (YMAX–YMIN) / 32767.0.

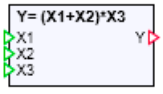
Parameter

Parameter name	Description	Unit	Default value
Formula	Formula that calculates output *y* in terms of inputs x_i.	–	X1

Example of a block with three inputs X1 to X3:

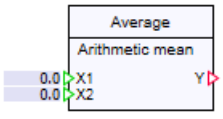
- (X1+X2)*X3

The formula is displayed at the top of the component symbol.



Average

Symbol



Function

The component type *Average* maps the average of the n inputs X_1 to X_n at output *Y*.

The number of inputs *n* is variable and can be set to any value between 2 and 32. The default value of all inputs is "0".

Arithmetic mean

The arithmetic mean is the sum of all inputs X_n divided by the number *n*.

$$Y = \frac{1}{n}\sum_{i=1}^{n} X_i = \frac{X_1 + X_2 + \cdots + X_n}{n}$$

Median

All inputs X_n are arranged in (ascending) order by value. The median is the value exactly in the middle.

$$Y = \begin{cases} x_{\frac{n+1}{2}} & n \text{ ungerade} \\ \frac{1}{2}\left(x_{\frac{n}{2}} + x_{\frac{n}{2}+1}\right) & n \text{ gerade} \end{cases}$$

Geometric mean

The geometric mean is the *n*th root of the product of all inputs X_n.

$$Y = \sqrt[n]{\prod_{i=1}^{n} X_i} = \sqrt[n]{X_1 \cdot X_2 \cdots X_n}$$

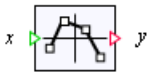
If the product is negative, the error message *"Average – geometric mean: the product of all inputs is negative"* (message category: *ERROR*) is generated and output *Y* is set to "0".

Parameter

Parameter name	Description	Unit	Default value
CalculationType	Type of mean; *Arithmetic mean*, *Median* or *Geometric mean*	–	*Arithmetic mean*

Characteristic

Symbol



Function

The Characteristic component type is used to define the mapping of input value *x* onto output value *y* defined by a characteristic:

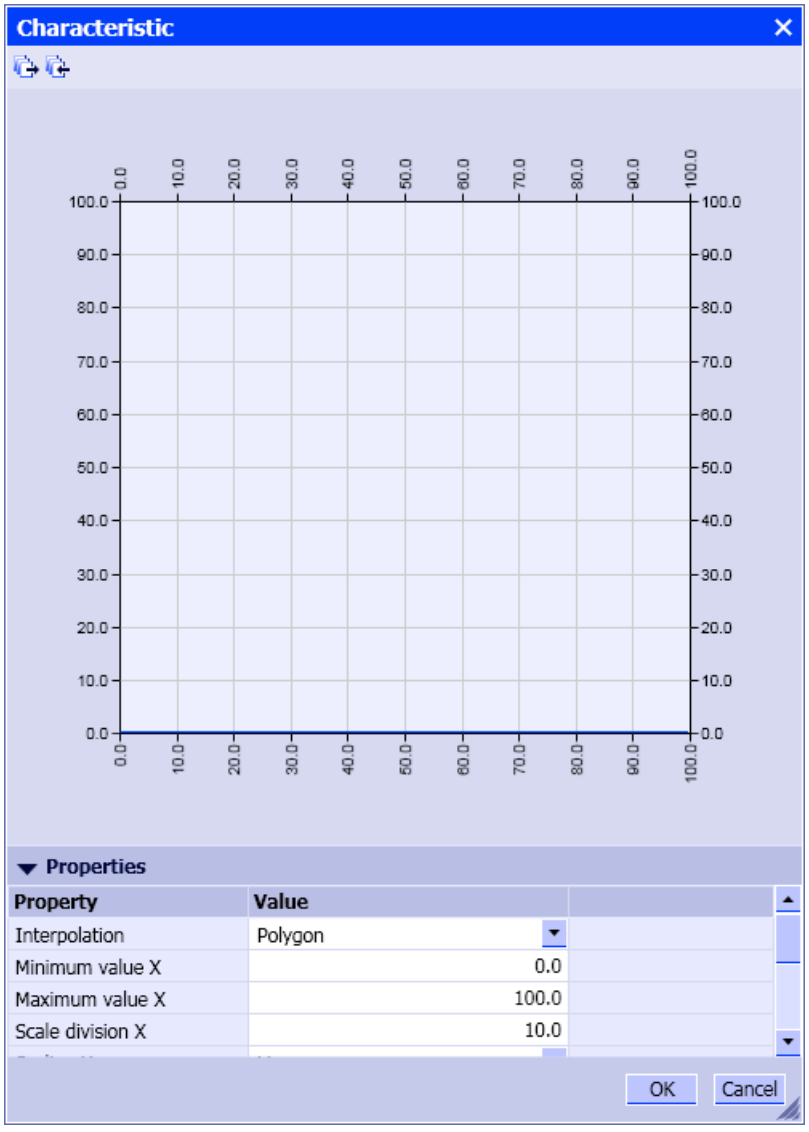
$y = f_K(x)$

Parameters

Parameter name	Description	Unit	Default value
Characteristic	Defines the characteristic to be used. Open the characteristic editor with .	–	–

Define characteristic

In this editor a characteristic is defined by specifying n interpolation points (x_i, y_i), $i = 1, ..., n$ and by selecting the type of *Interpolation* between these interpolation points.



There may be any number *n* of interpolation points. The type of interpolation may be either constant or linear, in which case, the characteristic will take the form of a *step curve* or a *polyline*. *Polyline* is the default mode.

Outside the interpolation point range (the interpolation range), the output values *y* are extrapolated.

For each of the two axes, which means for the horizontal x-axis and the vertical y-axis of the characteristic chart, the following variables can be set in the properties of the chart:

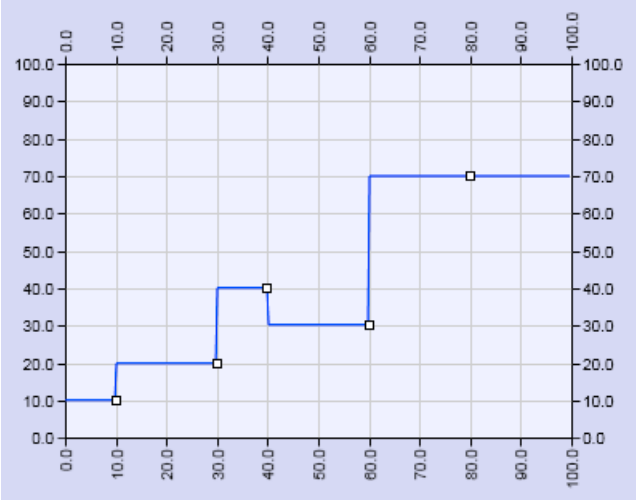
- the *minimum value* and the *maximum value*,
- the *scale division* and
- the *scale*.

Both axes default to a linear scale with a scale division of ten, a minimum value of zero and a maximum value of one hundred.

The *scale* may be set to either *linear* or *logarithmic*.

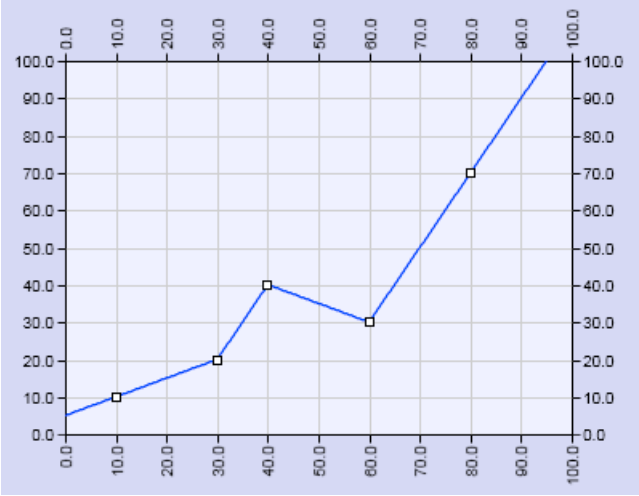
The characteristic function for *n* interpolation points is thus given as 0.0 for $n = 0$, y_1 for $n = 1$ and for $n > 1$ for the step curve using

$$y = \begin{cases} y_1 \ \text{for} \ x \le x_1, \\ y_i \ \text{for} \ x_{i-1} < x \le x_i, \ \ i = 2, \ldots, n \\ y_n \ \text{for} \ x > x_n \end{cases}$$



and for the polyline using

$$y = \begin{cases} y_1 + \dfrac{y_2 - y_1}{x_2 - x_1}(x - x_1) \ \text{for} \ x \le x_1 \\ y_{i-1} + \dfrac{y_i - y_{i-1}}{x_i - x_{i-1}}(x - x_{i-1}) \ \text{for} \ x_{i-1} < x \le x_i, \ i = 2, \ldots, n \\ y_n + \dfrac{y_n - y_{n-1}}{x_n - x_{n-1}}(x - x_n) \ \text{for} \ x > x_n \end{cases}$$



Inserting interpolation points:

To insert a interpolation point into the chart, move the cursor to the required position. The current coordinates are displayed at the cursor position. A new interpolation point is inserted at the current position with a double-click.

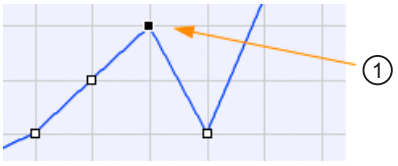




Alternatively, you can also insert an interpolation point at the current position by using the shortcut menu.

Removing interpolation points:

To remove an interpolation point, select it. The selected interpolation point changes its appearance.



① Selected interpolation point

You can remove the selected interpolation point with <Delete> or with the shortcut menu.

Changing the coordinates of interpolation points

Select a interpolation point with a left-click and by dragging it to the required position while keeping the left mouse button pressed. The interpolation point can only be moved within the boundaries defined by the x-coordinates of its left and right neighbors.

You can also change both coordinates of the selected interpolation point in the properties. Open the input by clicking in the input field, and then enter each desired value. Input is confirmed by pressing *Enter*.

Property	Value
X	50.0
Y	39.809553665896

Importing and exporting interpolation points

You can import interpolation points from a file. The command from the toolbar of the characteristic editor opens the dialog for selecting an Excel file in csv format. In this interpolation point file there is one interpolation point per line, given as pairs of values with their two coordinates: x-coordinate separator y-coordinate. If a dot "." is used as the decimal point, the separator is a comma ",". If a comma "," is used as the decimal point, the separator is a semicolon ";".

With the command from the toolbar of the characteristic editor, you can also export the current interpolation points into a csv file where you can modify them, for example. A file is written which contains all interpolation points as pairs of values (x-coordinate and y-coordinate). The decimal point is a comma ","; the separator is a semicolon ";".

Compare – Comparison functions

Symbol

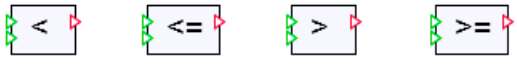


Function

The component type *Compare* compares the analog inputs x_1 and x_2. The binary output *b* is set to 1 if the relational expression is true – otherwise, *b* is set to 0.

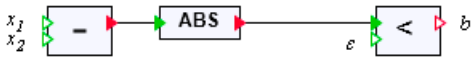
The selected comparison operator is displayed in the component symbol.

The width of the symbol can be changed to slightly offset the">=" and "<=" comparison operators from the right edge of the symbol.

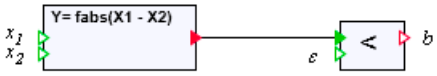


In SIMIT, analog variables are mapped to variables of the type double. A direct comparison for equality or inequality is therefore not useful. double variable equality is only possible within the

accuracy defined by the computer (machine accuracy). Equality can be checked using the model illustrated in the figure below, for example.



Alternatively, a functionally identical model can be used, for example with the *AFormula* formula block.



The difference between the two variables x_1 and x_2 is calculated. The amount of this difference is then compared against a definable positive limit ϵ:

$$b = \begin{cases} 1, \text{ if } & |x_1 - x_2| < \varepsilon \\ 0, \text{ otherwise} & \end{cases}$$

To check for inequality, only the "greater than" comparison should be used in the comparison component in both of the illustrated models.

Note

You can create your own components to check for equivalence or non-equivalence based on the explanatory notes provided above.

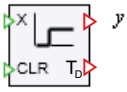
You can create macro components with the macro component editor, for example, or a relational component extended to include these comparisons with the Component Type Editor (CTE).

Parameter

Parameter name	Description	Unit	Default value
Comparison	Specifies the type of comparison: • Less than comparison (<), which means $b = \begin{cases} 1, \text{ if } & x_1 < x_2 \\ 0, \text{ otherwise} \end{cases}$ • Less than or equal to comparison (<=), which means $b = \begin{cases} 1, \text{ if } & x_1 \leq x_2 \\ 0, \text{ otherwise} \end{cases}$ • Greater than comparison (>), which means $b = \begin{cases} 1, \text{ if } & x_1 > x_2 \\ 0, \text{ otherwise} \end{cases}$ • Greater than or equal to comparison (>=), which means $b = \begin{cases} 1, \text{ if } & x_1 \geq x_2 \\ 0, \text{ otherwise} \end{cases}$	–	<

DeadTime – Dead time element

Symbol



Function

The *DeadTime* component type makes a dead time element available. The analog value at input *x* is passed to output *y* with an adjustable delay.

For this purpose, the same number of memory cells configured with the *Delay_Cycles* parameter are created in the component. These memory cells are preassigned with zero and can be set back to zero at any time using the binary input *CLR* . The input value is shifted through these memory cells in every computing cycle or every nth computing cycle. The last memory cell is output at output *y*.

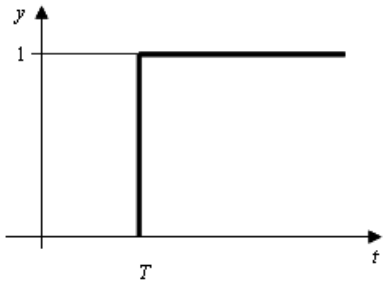
When *Sampling_Time* = 0 is configured, values are acquired and shifted forward in each cycle. The delay *T* is the product of "Delay_Cycles" and the current sampling time.

With parameter "Sampling_Time", in seconds, it is possible to extend the time scale in which the values of the component are acquired and shifted through. The actual delay *T* is then the product of "Delay_Cycles" and "Sampling_Time".

Note that this calculation only works out exactly if "Sampling_Time" is a multiple of the sampling time. That is because the effective sampling time is rounded to a multiple of the sampling time.

The actual delay in seconds, after which the change at input *X* shows up at output *Y* is available at output T_D.

A step change of input value *x* from zero to one thus yields the curve of output value *y* as shown in the figure below:



Note

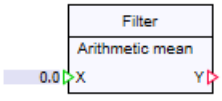
If you change the cycle time Δ*t* in the project properties, the set dead times change accordingly.

Parameter

Parameter name	Description	Unit	Default value
Delay_Cycles	Number of memory cells through which the input signal is shifted. Maximum value: 128	–	10
Sampling_Time	Minimum amount of simulation time that must have elapsed before the current input signal is accepted again.	s	0.0

Filter

Symbol



Function

The *Filter* component type maps the average over time of the *X* input to output *Y*.

Arithmetic mean

The arithmetic mean is calculated by adding up the time values of the input *X*n and dividing it by the number of cycles *n*.

$$Y = \frac{1}{n}\sum_{i=0}^{n-1} X_{n-i} = \frac{X_n + X_{n-1} + \cdots + X_{n-(n-1)}}{n}$$

Median

All time values of the input *X*n are put in (ascending) order by value. The median is the value exactly in the middle.

$$Y=\begin{cases} x_{\frac{n+1}{2}} & n\ ungerade \\ \frac{1}{2}\left(x_{\frac{n}{2}}+x_{\frac{n}{2}+1}\right) & n\ gerade \end{cases}$$

Geometric mean

The geometric mean is the *n*th root of the product of all times values of the input *X*.

$$Y=\sqrt[n]{\prod_{i=0}^{n-1}X_{n-i}}=\sqrt[n]{X_n\cdot X_{n-1}\cdots X_{nn-(n-)}}$$

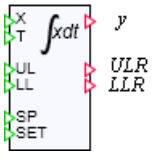
If the product is negative, the error message "*Filter – geometric mean: the product Xi is negative*" (message category: *ERROR*) is generated and output *Y* is set to "0".

Parameter

Parameter name	Description	Unit	Default value
CalculationType	Calculation method for the mean	–	*Arithmetic mean*
Number_Cycles	Number of cycles considered Value range: 1 to 128	–	3

INT – Integration

Symbol



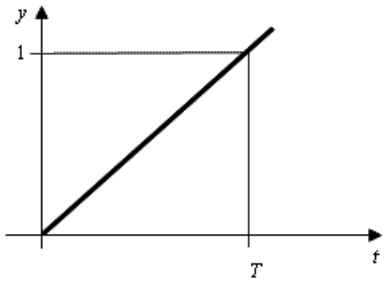
Function

The *INT* component type forms the integral with the time-specific analog input signal *x*:

$$y=\frac{1}{T}\int x\ \mathrm{d}t$$

The integral value is assigned to the analog output *y*.

For a step-shaped input signal of amplitude 1, the result is a linear ascending output signal as shown in the figure below.



The integral value *y* is limited to an interval defined by the two limits *UL* (high limit) and *LL* (low limit):

$LL \le y \le UL$.

The binary outputs *ULR* and *LLR* indicate that the integral value has reached the low or high limit:

$$ULR = \begin{cases} 1, \text{ if } & y \ge UL \\ 0, \text{ otherwise} \end{cases}$$

and

$$LLR = \begin{cases} 1, \text{ if } & y \le LL \\ 0, \text{ otherwise} \end{cases}$$

The low limit must be less than the high limit. If this condition is violated, the error message "*INT: limits do not match*" (message category *ERROR*) is generated and output *y* is set to 0.

The time constant *T* for the integration must have a positive value. If *T* is not positive, the error message "*INT: zero or negative time constant*" (message category *ERROR*) is generated and output *y* is set to zero.

The default setting is 0 for the low limit and 100 for the high limit. The default setting for the time constant *T* is 1 s. All other inputs are set to 0 by default.

The integration value *y* can be set to the value at input *SP* over the binary input *SET*: if *SET* is set to one, the integration value *y* is set to the value at the *SP* input. In this case too, the integration value is limited by *LL* and *UL*.

Parameter

Parameter name	Description	Unit	Default value
Initial_Value	Initial value of the integral value *y* when starting the simulation	–	0.0

Operating window

The operating window shows the current function value as a percentage.

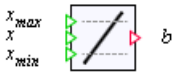
You can change the following values:

- Input value *X* even during the simulation
- Setpoint SP. Confirm the value with SET.

When the inputs *SP*, *SET* or *X* are connected to other components with signal lines, you must first enable the respective signal isolator for manual operation.

Interval – Interval query

Symbol



Function

The *Interval* component type checks whether an input value *x* is within the closed interval [x_{min}, x_{max}]. If the input value is within the specified interval, the binary output is set to one; otherwise, it is set to zero:

$$b = \begin{cases} 1, \text{ for } & x_{min} \le x \le x_{max} \\ 0, \text{ otherwise} & \end{cases}$$

Parameter

Parameter name	Description	Unit	Default value
x_{min}	Low interval limit	–	0
x_{max}	High interval limit	–	100

The high interval limit may not be less than the lower interval limit. If the high interval limit becomes less than the low limit during the simulation, the error message "*Interval_I: limits do not match*" (message category *ERROR*) is generated and output *b* is set to zero.

Limiter

Symbol



Function

The *Limiter* component type maps an input value limited to the range from x_{min} to x_{max} onto the output *y*:

$$y = \begin{cases} x_{max} \text{ for } x \geq x_{max} \\ x \quad \text{for } x_{min} < x < x_{max} \\ x_{min} \text{ for } x \leq x_{min} \end{cases}$$

The binary outputs b_{min} and b_{max} are set to one when the limiter takes effect, which means

$$b_{max} = \begin{cases} 1, \text{ if} & x \geq x_{max} \\ 0, \text{ otherwise} \end{cases}$$

and

$$b_{min} = \begin{cases} 1, \text{ if} & x \leq x_{min} \\ 0, \text{ otherwise} \end{cases}$$

Parameter

Parameter name	Description	Unit	Default value
x_{min}	Low limit	–	0
x_{max}	High limit	–	100

The high limit must not be less than the low limit. If the low limit is equal to or greater than the high limit when simulation is run, the error message "*Limiter: limits do not match*" (message category *ERROR*) is generated and output *y* is set to zero.

MinMax – minimum and maximum value selection

Symbol



Function

The component type *MinMax* maps the minimum or maximum of the *n* inputs X_1 to X_n to output *y*. The number of inputs *n* is variable and can be set to any value between 2 and 32. The default value of all inputs is 0.

The *MIN* or *MAX* mapping set is displayed in the component symbol as shown in the figure below:

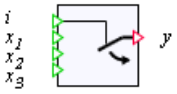


Parameter

Parameter name	Description	Unit	Default value
MinMax	Type of extreme value to be calculated	–	*MIN*

Multiplexer

Symbol



Function

The component type *Multiplexer* interconnects one of the *n* inputs x_i to output in *y* in line with the value at the integer selection input *i*:

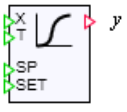
$y = x_i$ for $1 \le i \le n$.

The number *n* of inputs x_i is variable and can be set to a value of between 2 and 32. The default value of all inputs is "0".

The value at the selection input *i* is limited internally to 1 through *n*, which means that *i* is set to "1" or values less than 1 and *i* is set to *n* for values greater than *n*. In the default setting, the first input x_i is therefore interconnected with the output *y*.

PTn – nth order delay

Symbol



Function

The component type *PTn* provides an nth order delay.

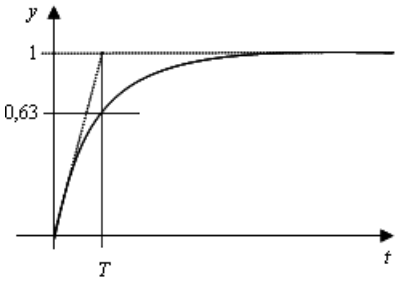
With a first order delay, the function value *y* at the output follows the value *x* at the input with a delay equal to the differential equation:

$$\frac{dy}{dt} = \frac{1}{T}(x - y)$$

A step change in the input value *x* from 0 to 1 therefore results in an exponential curve for the output value *y* (step response):

$y = 1 - e^{-t/T}$

as illustrated in the figure below.



For higher-order delays, the output value *y* is obtained by concatenating first-order delays:

$$\frac{dz_i}{dt} = \frac{1}{T}(z_{i-1} - z_i),\ i = 1, \ldots, n$$

$y = z_n$

Where z_i, $i = 1,...n$, are the *n* states of the *n*th order delay. To illustrate, an *n*th order delay corresponds to *n* first order delays connected in series.

To prevent the component behaving in an unstable manner for delay times that are too short, the delay time constant *T* is limited to values that are greater than or equal to the cycle time of the component. If the values at input *T* are less than the cycle time, the error message "*PTn: delay time below cycle time*" (message category *ERROR*) is generated. The default delay time is 1 s. All other inputs of the component are set to 0 by default.

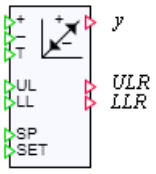
The binary input *SET* may be used to set the output value y and the state values z_i, $i = 1,...n$, to the value at the input *SP*: If *SET* is set to 1, these values are set to the value at the input *SP*.

Parameter

Parameter name	Description	Unit	Default value
n	Order of the delay Maximum value: 32	–	1
Initial_Value	Initial value for the delay function states	–	0.0

Ramp – Ramp function

Symbol



Function

The component type *Ramp* increments or decrements its function value *y* in each increment of the simulation by the value

$$\Delta y = (UL - LL)\frac{\Delta t}{T}$$

where Δ*t* is the simulation increment width. The ramp value *y* is incremented by Δ*y* when the "+" (*UP*) input has the value 1. If the "–" input *DOWN* has the value 1, the ramp value *y* is decremented by Δ*y*. If both the "+" and the "–" input are set to 1, the ramp value *y* does not change.

The ramp value is limited to an interval defined by the two limits *UL* (high limit) and *LL* (low limit):

$LL \le y \le UL$.

The binary outputs *ULR* and *LLR* indicate that the ramp value has reached the low or high limit:

$$ULR = \begin{cases} 1, \text{ if } & y \ge UL \\ 0, \text{ otherwise} \end{cases}$$

and

$$LLR = \begin{cases} 1, \text{ if } & y \le LL \\ 0, \text{ otherwise} \end{cases}$$

The low limit must be less than the high limit. If this condition is violated, the error message "*RAMP: limits do not match*" (message category *ERROR*) is generated and output *y* is set to 0.

The time constant *T* must have a positive value. If *T* is not positive, the error message "*Ramp: zero or negative time constant*" (message category *ERROR*) is generated and output *y* is set to 0.

The default low limit is 0 and the default high limit 100. The time constant is set to 10 seconds by default. All other inputs are set to 0 by default.

The binary input *SET* can be used to set the ramp value *y* to the value at the *SP* input: If *SET* is set to 1, the ramp value *y* is set to the value at the input *SP*. Here too, the ramp value is limited by *LL* and *UL*.

Parameter

Parameter name	Description	Unit	Default value
Initial_Value	Initial value of the ramp function	–	0.0

Operating window

The operating window shows the current function value as a percentage.

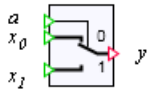
You can change the following values:

- Ramp function value over "Down" and "Up" buttons even during simulation
- Setpoint SP. Confirm the value with SET.

When the inputs *SP*, *SET*, "Down" or "Up" are connected to other components with signal lines, you must first enable the respective signal isolator for manual operation.

Selection – Analog switch

Symbol



Function

The component type *Selection* interconnects one of the two inputs x_0 and x_1 to output y in line with the value of the binary input *a*.

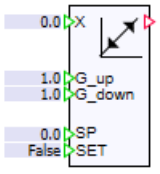
If the selection input $a = 0$, input x_0 is interconnected; if the selection input $a = 1$, input x_1 is interconnected:

$$y = \begin{cases} x_0, \text{ if } & a = 0 \\ x_1, \text{ if } & a = 1 \end{cases}$$

The default value of all inputs is 0.

Tracking

Symbol



Function

The component type *Tracking* tracks the outputs value *Y* to input value *X* with an adjustable gradient. You can set the positive gradient (*G_up*) and negative gradient (*G_down*) separately. The unit of the gradients is 1/second. If the *Y* output corresponds to the *X* input, the *Y* output value remains constant.

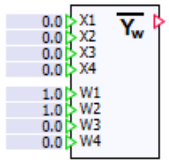
The binary input *SET* can be used to set the output value *Y*to the value at the *SP* input. If *SET* is set to True, the output value *Y* is set to the value at the input *SP*.

Parameter

Parameter name	Description	Unit	Default value
Initial_Value	Initial value of the tracking function	–	0.0

WeightedAverage – Weighted arithmetic mean

Symbol



Function

The component type *Weighted Average* maps the weighted arithmetic mean of the analog inputs X_1, X_2, X_3 and X_4 at output *Y* with the weightings W_1, W_2, W_3 and W_4.

$$Y = \frac{X_1 W_1 + X_2 W_2 + X_3 W_3 + X_4 W_4}{W_1 + W_2 + W_3 + W_4}$$

The sum of W_1, W_2, W_3 and W_4 must not be "0". If the sum is equal to "0" when simulation is run, the error message "*Weighted_Average:sum of W_1, W_2, W_3 and W_4 is zero – division by zero*" (message category*ERROR*) is generated and output *Y* is set to "0".

The default setting for inputs X_1, X_2, X_3, X_4, W_3 and W_4 is "0". The default setting for inputs W_1 and W_2 is 1.0.

8.1.3.3 Integer functions

Integer functions as component types

All component types with integer functions are contained in the *IntegerBasic* and *IntegerExtended* directories of the standard library.

The four basic integer functions of Addition, Subtraction, Multiplication and Division, which means the four basic arithmetic operations, are stored as component types in the *IntegerBasic* directory of the standard library.

The symbols of these component types are blue to distinguish them from analog component types.

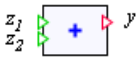
The *IntegerExtended* directory of the standard library contains further integer functions in the form of component types.

The symbols of these component types are blue to distinguish them from analog component types.

Basic integer functions

ADD_I – Addition

Symbol



Function

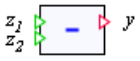
The *ADD_I* component type maps the sum of the analog values at the *n* inputs x_1 to x_n onto the output *y*:

$$y = \sum_{i=1}^{n} x_i = x_1 + x_2 + \ \ldots \ + x_n$$

The number of inputs *n* is variable and can be set to any value between 2 and 32. All inputs are set to zero by default.

SUB_I – Subtraction

Symbol



Function

The *SUB_I* component type maps the difference between the integer values at the two inputs x_1 and x_2 onto the output *y* according to

$y = x_1 - x_2$

All inputs are set to zero by default.

MUL_I – Multiplication

Symbol



Function

The *MUL_I* component type maps the product of the integer values at the *n* inputs x_1 to x_n onto the output *y*:

$$y = \prod_{i=1}^{n} x_i = x_1 x_2 \ \ldots \ x_n$$

The number of inputs *n* is variable and can be set to any value between 2 and 32. All inputs are set to one by default.

DIV_I – Integer division

Symbol



Function

The *DIV_I* component type maps the quotients of the integral values at the inputs x_1 and x_2 according to

$x_1 = y \cdot x_2 + R$

integer division with remainder. The integer quotient of the division is output at the *y* output and the remainder of the integer division at the *R* output.

The divider at input x_1 is set to zero by default, while the divisor input x_2 is set to one by default.

The value of the divisor x_2 must not be zero. If the devisor becomes zero during the simulation, the error message "*DIV_I: division by zero*" (message category *ERROR*) is generated and output *y* and the remainder *R* are set to zero.

Extended integer functions

Compare_I – Compare functions

Symbol



Function

The component type *Compare_I* compares the integer inputs x_1 and x_2. The binary output *b* is set to 1 if the relational expression is true. Otherwise, *b* is set to 0.

The selected comparison operator is displayed in the component symbol. To inset the relational operators "<=", ">=" and "<>" from the right of the symbol, change the width of the symbol.

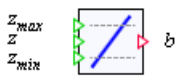


Parameters

Parameter name	Description	Unit	Default value
Comparison	Specifies the type of comparison: • Less than comparison (<), which means $b = \begin{cases} 1, \text{ if } & x_1 < x_2 \\ 0, \text{ otherwise} \end{cases}$ • Less than or equal to comparison (<=), which means $b = \begin{cases} 1, \text{ if } & x_1 \leq x_2 \\ 0, \text{ otherwise} \end{cases}$ • Greater than comparison (>), which means $b = \begin{cases} 1, \text{ if } & x_1 > x_2 \\ 0, \text{ otherwise} \end{cases}$ • "Greater than or equal to" comparison (>=), which means $b = \begin{cases} 1, \text{ if } & x_1 \geq x_2 \\ 0, \text{ otherwise} \end{cases}$ • Equal to comparison (=), which means $b = \begin{cases} 1, \text{ if } & x_1 = x_2 \\ 0, \text{ otherwise} \end{cases}$ • Not equal to comparison (<>), which means $b = \begin{cases} 0, \text{ if } & x_1 \neq x_2 \\ 1, \text{ otherwise} \end{cases}$	–	<

Interval_I – Interval query

Symbol



Function

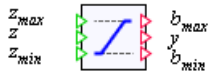
The *Interval_I* component type checks whether an integer input value *x* is within the closed interval [x_{min}, x_{max}]. If the input value falls within the set interval, the binary output is set to one; otherwise it is zero:

$$b = \begin{cases} 1, \text{ for } & x_{min} \le x \le x_{max} \\ 0, \text{ otherwise} & \end{cases}$$

The high interval limit may not be less than the lower interval limit. If the high interval limit becomes less than the low limit during the simulation, then the error message "*Interval_I: limits do not match*" (message category *ERROR*) is generated and output *b* is set to zero.

Limiter_I – Limiter

Symbol



Function

The *Limiter_I* component type maps an input value limited to the range from x_{min} to x_{max} onto the output *y*:

$$y = \begin{cases} x_{max} \text{ for } & x \ge x_{max} \\ x \text{ for } & x_{min} < x < x_{max} \\ x_{min} \text{ for } & x \le x_{min} \end{cases}$$

The binary outputs b_{min} and b_{max} are set to one when the limiter takes effect, which means

$$b_{max} = \begin{cases} 1, \text{ if } & x \ge x_{max} \\ 0, \text{ otherwise} & \end{cases}$$

and

$$b_{min} = \begin{cases} 1, \text{ if } & x \le x_{min} \\ 0, \text{ otherwise} & \end{cases}$$

The high limit must not be less than the low limit. If the low limit is greater than or equal to the high limit when simulation is run, the error message "*Limiter_I: limits do not match*" (message category *ERROR*) is generated and output *y* is set to zero.

MinMax_I – Minimum and maximum value selection

Symbol



Function

The component type *MinMax_I* maps the minimum or maximum of the *n* inputs z_1 to z_n to output *y*. The number of inputs *n* is variable and can be set to any value between 2 and 32. The default value of all inputs is 0.

The *MIN* or *MAX* mapping set is displayed in the component symbol as shown in the figure below:

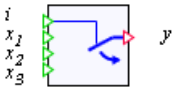


Parameter

Parameter name	Description	Unit	Default value
MinMax	Type of extreme value to be calculated	–	*MIN*

Multiplexer_I – Integer multiplexer

Symbol



Function

The component type *Multiplexer_I* interconnects one of the *n* different Integer inputs x_i to output in *y* in line with the value at the integer selection input *i*:

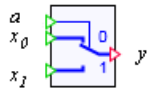
$y = x_i$ for $1 \le i \le n$.

The number *n* of inputs x_i is variable and can be set to a value of between 2 and 32. The default value of all inputs is 0.

The value at the selection input *i* is limited internally to 1 through *n*, which means that *i* is set to 1 or values less than 1 and *i* is set to *n* for values greater than *n*. In the default setting, the first input x_i is therefore interconnected with the output *y*.

Selection_I – Integer switch

Symbol



Function

The component type *Selection_I* interconnects one of the two integer inputs x_0 and x_1 to output *y* in line with the value of the binary input *a*.

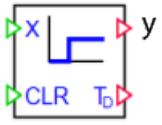
If the selection input $a = 0$, input x_0 is interconnected; if the selection input $a = 1$, input x_1 is interconnected:

$$y = \begin{cases} x_0, \text{ if } & a = 0 \\ x_1, \text{ if } & a = 1 \end{cases}$$

The default value of all inputs is 0.

DeadTime_I – Dead time element

Symbol



Function

The *DeadTime_I* component type makes a dead time element available. The integer value at input *x* is passed to output *y* with a configurable delay.

For this purpose, the same number of memory cells configured with the *Delay_Cycles* parameter are created in the component. These memory cells are preassigned with zero and can be set back to zero at any time using the binary input *CLR* . The input value is shifted through these memory cells in every computing cycle or every nth computing cycle. The last memory cell is output at output *y*.

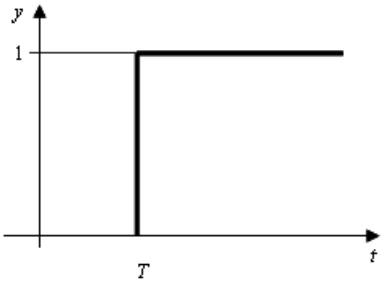
When *Sampling_Time* = 0 is configured, values are acquired and shifted forward in each cycle. The delay *T* is the product of "Delay_Cycles" and the current sampling time.

With parameter "Sampling_Time", in seconds, it is possible to extend the time scale in which the values of the component are acquired and shifted through. The actual delay *T* is then the product of "Delay_Cycles" and "Sampling_Time".

Note that this calculation only works out exactly if "Sampling_Time" is a multiple of the sampling time. That is because the effective sampling time is rounded to a multiple of the sampling time.

The actual delay in seconds, after which the change at input *X* shows up at output *Y* is available at output T_D.

A step change of input value *x* from zero to one thus yields the curve of output value *y* as shown in the figure below:



Note

If you change the cycle time Δ*t* in the project properties, the set dead times change accordingly.

Parameter

Parameter name	Description	Unit	Default value
Delay_Cycles	Number of memory cells through which the input signal is shifted. Maximum value: 128	–	10
Sampling_Time	Minimum amount of simulation time that must have elapsed before the current input signal is accepted again.	s	0.0

8.1.3.4 Mathematical functions

Mathematical functions as component types

The *Math* directory of the Standard Library contains the most commonly used mathematical functions in the form of component types: absolute value generation (*ABS*), square root extraction (*SQRT*), natural logarithm (*LN*) and the exponential function (*EXP*), plus the trigonometrical functions sine (*SIN*), cosine (*COS*) and tangent (*TAN*).

Note

You can also use the *AFormula* component with suitable parameters instead of these components.

You can increase the available mathematical functions of this type by creating suitable component types using the SIMIT Component Type Editor (CTE) expansion module.

ABS – Absolute value

Symbol



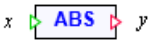
Function

The *ABS* component type maps the absolute value of input *x* onto output *y*:

$y = |x|$.

ABS_I – Absolute integer values

Symbol



The symbol for *ABS_I* is blue to distinguish it from the *ABS* component type.

Function

The *ABS_I* component type maps the absolute value of integer input *x* onto integer output *y*:

$y = |x|$.

COS – Cosine function

Symbol



Function

The component type *COS* maps the cosine of input *x* to output *y*:

$y = \cos(x)$.

Parameter

Parameter name	Description	Unit	Default value
Unit	Unit of measurement of argument *x*: • rad (radians) • deg (degrees)	–	rad

EXP – Exponential function

Symbol



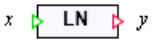
Function

The *EXP* component type maps the exponential value of input *x* onto output *y*:

$y = e^x$

LN – Natural logarithm

Symbol



Function

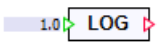
The *LN* component type maps the natural logarithm of input *x* onto output *y*:

$y = \ln(x)$.

The argument *x* must be positive. If the argument becomes less than or equal to zero during the simulation, the error message "*LN: invalid argument*" (message category *ERROR*) is generated and output *y* is set to zero. The argument *x* is set to one by default.

LOG – Common logarithm

Symbol



Function

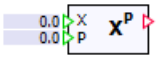
The *LOG* component type maps the common logarithm of the *X* input to output *Y*.

$Y = Log_{10}X$

Argument *X* must be positive. If the argument is less than or equal to "0" when simulation is run, the error message "*LOG: invalid argument*" (message category *ERROR*) is generated and output *Y* is set to "0". The default setting for input *X* is 1.0.

POW – Power

Symbol



Function

The *POW* component type maps the base (input) *X* to the power (input) P to output *Y*.

$Y = X^{P}$

The default value of all inputs is 0.0.

RAND – random numbers

Symbol



Function

The component type RAND returns randomly generated numbers at its output Y, the ranges and distribution of which are set with parameter *Distribution*.

Note

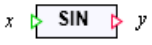
Note that the random numbers are calculated and therefore are not unpredictable.

Parameter

Parameter name	Description	Unit	Default value
Distribution	Value range and distribution of random numbers • *Uniform distribution* The random numbers are equally distributed in the open interval (0,1). • *Normal distribution* The random numbers are normally distributed; the mean is 0 and the standard deviation 1.	–	*Uniform distribution*

SIN – Sine function

Symbol



Function

The component type *SIN* maps the sine of input *x* to output *y*:

$y = \sin(x)$.

Parameter

Parameter name	Description	Unit	Default value
Unit	Unit of measurement of argument *x*: • rad (radians) • deg (degrees)	–	rad

SQRT – Square root

Symbol



Function

The *SQRT* component type maps the square root of input *x* onto output *y*:

$y = \sqrt{x}$

The radicand *x* must not be negative. If the radicand becomes negative during the simulation, the error message "*SQRT: invalid argument*" (message category *ERROR*) is generated and output *y* is set to zero.

TAN – Tangent function

Symbol



Function

The component type *TAN* maps the tangent of input *x* to output *y*:

$y = \tan(x)$.

Parameter

Parameter name	Description	Unit	Default value
Unit	Unit of measurement of argument *x*: • rad (radians) • deg (degrees)	–	rad

8.1.3.5 Binary functions

Binary functions as component types

All of the functions for processing binary signals are contained in the *BinaryBasic* and *BinaryExtended* directories.

Component types with the three basic binary operations of conjunction (*AND*), disjunction (*OR*) and negation (*NOT*), plus equivalence (*XNOR*) and non-equivalence (*XOR*) are stored in the *BinaryBasic* directory.

The *BinaryExtended* directory of the standard library contains further binary (logic) functions that go beyond the elementary binary functions in the form of component types.

Basic binary functions

AND – Conjunction

Symbol



Function

The *AND* component type maps the *n* binary values at the inputs a_i as a conjunction, which means using a logical (Boolean) "AND" function, onto the output *b*:

$$b = \bigwedge_{i=1}^{n} a_i$$

The number of inputs *n* is variable and can be set to any value between 2 and 32. All inputs are set to one by default.

OR – Disjunction

Symbol



Function

The *OR* component type maps the *n* binary values at the inputs a_i as a disjunction, which means using a logical (Boolean) "OR" function, onto the output *b*:

$$b = \bigvee_{i=1}^{n} a_i$$

The number of inputs *n* is variable and can be set to any value between 2 and 32. All inputs are set to zero by default.

NOT, NOTc – Negation

Negation is available in two component types *NOT* and *NOTc*. These differ only in terms of the symbols used – their functions are completely identical.

Symbol



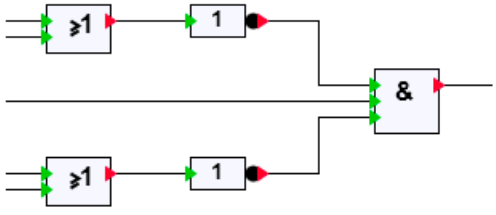
Function

Output b is equal to the negated input a, which means

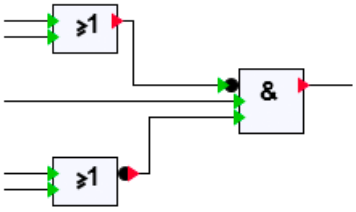
$$b = \overline{a}$$

As can be seen in the figure below, when the *NOTc* component type is used, the negation can be applied clearly and compactly to the inputs or outputs of components.

Using "NOT"

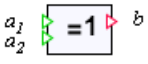


Using "NOTc"



XOR – Non-equivalence

Symbol



Function

The *XOR* component type maps the *n* binary values at the inputs a_i onto the output *b* using the "Exclusive-OR" function: *b* is one, if an odd number of inputs a_i is one. In all other cases *b* is zero.

For two inputs

$b = a_1 \otimes a_2$

which is the non-equivalence or "unequal" function.

The number of inputs *n* is variable and can be set to any value between 2 and 32. All inputs are set to zero by default.

XNOR – Equivalence

Symbol



Function

The *XNOR* component type maps the *n* binary values at the inputs a_i onto the output *b* using the "Exclusive NOR" function: *b* is one, if an even number of inputs a_i is one. In all other cases *b* is zero.

For two inputs

$$b = \overline{a_1 \otimes a_2}$$

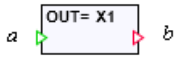
which is the equivalence or "equal" function.

The number of inputs *n* is variable and can be set to any value between 2 and 32. All inputs are set to zero by default.

Extended binary functions

BFormula – Binary formula component

Symbol



Function

The component type *BFormula* allows the use of explicit logical expressions. This logic function *f* calculates a value in relation to the *n* values at the inputs a_i. The function value is assigned to the output *b*:

$b = f(a_1,...,a_n)$

The number *n* of inputs can be varied between 1 and 32. Only the inputs that are available according to the number currently set may be used in the formula.

The operators listed in the following table may be used in formulas.

Table 8-3 Permitted operators in formulas for the BFormula component

Operator	Function
AND	Conjunction (AND function)
OR	Disjunction (OR function)

Operator	Function
NOT	Negation
(	Opening parenthesis
)	Closing parenthesis

Note

There is no check in the formula component to determine whether all the set inputs are used in the specified formula.

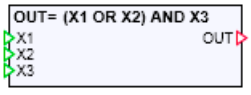
Parameter

Parameter name	Description	Unit	Default value
Formula	formula that calculates the output value *b* in terms of the input values a_i.	–	X1

Example of a block with three inputs X1 to X3:

- (X1 OR X2) AND X3

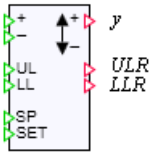
The formula is displayed at the top of the component symbol.



You can make the component wider to allow longer formulas to be displayed in full.

Counter – Up and down counters

Symbol



Function

The component type *Counter* allows you to count the change in binary signals. When the binary value at the "+" input changes from zero to one, the counter value is incremented at output *y*. When the binary value at the "–" input changes from zero to one, the counter value is decremented at the *y* output.

The counter value is limited to an interval defined by the two limits *UL* (high limit) and *LL* (low limit):

$LL \leq y \leq UL$.

The binary outputs *ULR* and *LLR* indicate when the counter value has reached the low or high limit:

$$ULR = \begin{cases} 1, \text{ if } & y \geq UL \\ 0, \text{ otherwise} \end{cases}$$

and

$$LLR = \begin{cases} 1, \text{ if } & y \leq LL \\ 0, \text{ otherwise} \end{cases}$$

The low limit must be less than the high limit. If this condition is violated, the error message "*Counter: limits do not match*" (message category *ERROR*) is generated and the counter value at output *y* is set to zero.

The binary input *SET* may be used to set the counter value *y* to the value at the input *SP*: If *SET* is set to one, the counter value *y* is set equal to the value at the input *SP*. Here, too, the counter value is limited by *LL* and *UL*.

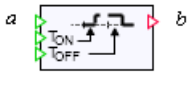
The limits are set to zero for the low limit and to one hundred for the high limit by default. All other inputs are set to zero by default.

Parameter

Parameter name	Description	Unit	Default value
Decrement Modifiable online	Value by which the counter value is decremented each time the binary value at input "–" changes	–	1.0
Increment Modifiable online	Value by which the counter value is incremented each time the binary value at the "+" input changes.	–	1.0
Initial_Value	Counter value when simulation is initialized (started)	–	1.0
TriggerUp	Type of edge change for incrementing the counter value	–	*Rising edge*
TriggerDown	Type of edge change for decrementing the counter value	–	*Rising edge*

Delay – On-Off delay

Symbol



Function

With the component type *Delay*, the binary signal *b* at the output is tracked to the binary signal *a* at the input with a delay.

If the signal at the input changes from zero to one, the output is set to one once the ON-delay time T_{ON} has elapsed. If the input signal is reset to zero before the ON-delay time has elapsed, the output signal remains unchanged at zero. If the signal at the input changes from one to zero, the output is set to zero once the OFF-delay time T_{OFF} has elapsed. If the output signal is reset to one before the OFF-delay time has elapsed, the output signal remains unchanged at one. This interaction is illustrated in the table below.

The unit for the ON and OFF times is seconds.

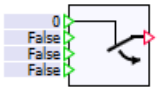
Table 8-4 Signal curves at the input and output of the Delay component

	T_{ON} T_{OFF}
Input a	
Output b	

The delay times must not assume a negative value. If a delay time assumes a negative value, the error message "*Delay: on-delay time negative value*" or "*Delay: off-delay time negative value*" (message category *ERROR*) is generated and the corresponding delay time is set to zero.

Multiplexer_B – Binary multiplexer

Symbol



Function

The component type *Multiplexer_B* interconnects one of the n inputs X_i to output *Y* in line with the value at the integer selection input *SEL*:

$Y = X_i$ *for* $1 \leq i \leq n$

The number n of inputs X_i is variable and can be set to a value of between 2 and 32. The default value of all inputs is False.

The value at the selection input *SEL* is limited internally to 1 through n. i is set to 1 for values less than 1 and *SEL* is set to n for values greater than n. In the default setting, the first input X_1 is interconnected with output *Y*.

Pulse – Pulse

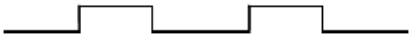
Symbol



Function

The component type *Pulse* sets output *b* if there is an edge change from 0 to 1 at input *a*. The output *b* is reset once the time *T* has elapsed. A pulse with pulse width *T* is thus generated at output *b*. The pulse width can be set via the signal at analog input *T*.

Table 8-5 Signal curves at the input and output of the Pulse component

	
Input a	
Output b	

The pulse width *T* must not be negative, or the error message "*Pulse: pulse width negative value*" (message category *ERROR*) is generated and output *b* is set to 0. For pulse widths that are smaller than the simulation cycle time, the output pulse is set to the duration of one simulation cycle.

Parameter

Parameter name	Description	Unit	Default value
Trigger Modifiable online	Type of edge change evaluated • *Rising edge*: an edge change from "False" to "True" (rising edge) sets output *b* to "True". • *Falling edge*: an edge change from "True" to "False" (falling edge) sets output *b* to "True".	–	*Rising edge*
Retriggerable Modifiable online	Type of response to input signal change from "0" to "1" • "True": the output pulse is started again each time the input signal changes from "0" to "1". • "False": the output pulse is not started again when the input signal changes from "0" to "1".	–	*False*

RS_FF – Flip-flop with default state of "Reset"

Symbol



Function

The component type *RS_FF* provides the simplest type of flip-flop: an RS flip-flop. If the value at the set input *S* is equal to one, the output value *Q* is set to one. If the input value at the reset input *R1* is set to one, the output is reset to zero. The reset input is dominant, which means if both inputs are set to one, output *Q* is reset to zero. The output $\overline{Q}$ always assumes the inverse value of output *Q*.

The table below lists the possible states of the *RS_FF* component type.

Table 8-6 State table for the RS_FF component

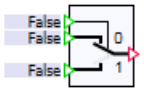
Input S	Input R1	Output Q	Output $\overline{Q}$
0	0	Unchanged	Unchanged
0	1	0	1
1	0	1	0
1	1	0	1

Parameter

Parameter name	Description	Unit	Default value
Initial_State	Initial value of output *Q*	–	0

Selection_B – Binary switch

Symbol



Function

The component type *Selection_B* interconnects one of the two binary inputs X_0 and X_1 to output *Y* in line with the value of the binary input *SEL*.

If selection input *SEL* = False, input X_0 is interconnected; if selection input *SEL* = True, input X_1 is interconnected.

$$Y = \begin{cases} X_0, \text{ if } SEL = false \\ X_1, \text{ if } SEL = true \end{cases}$$

The default value of all inputs is False.

SR_FF – Flip-flop with default state of "Set"

Symbol



Function

The component type *SR_FF* maps an SR flip-flop. If the value at the set input *S1* is equal to one, the output value *Q* is set to one. If the input value at the reset input *R* is set to one, the output is reset to zero. The set input is dominant, which means if both inputs are set to one, output *Q* is reset to one. The output $\overline{Q}$ always has the inverse value of output *Q*.

The table below lists the possible states of the component *SR_FF* .

Table 8-7 State table for the SR_FF component

Input S	Input R1	Output Q	Output $\overline{Q}$
0	0	Unchanged	Unchanged
0	1	0	1
1	0	1	0
1	1	1	0

Parameter

Parameter name	Description	Unit	Default value
Initial_State	Initial value of output *Q*	–	0

8.1.3.6 Converting values

Converting signals as component types

The *Conv* directory of the Standard Library contains component types for converting signals:

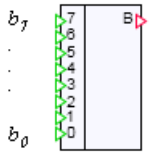
- *Bit2Byte* for converting bits to a byte value
- *Byte2Bit* for converting bytes to bits
- *Byte2Word* for converting bytes to a word
- *Word2Byte* for converting a word into bytes

- *Byte2DWord* for converting bytes to a double word
- *DWord2Byte* for converting a double word into bytes
- *Analog2Integer* for converting an analog value into an integer value
- *Integer2Analog* for converting an integer value into an analog value
- *Raw2Phys* for converting a raw value to an analog value
- *Phys2Raw* for converting an analog value to a raw value
- *Unsigned2Signed* for converting an unsigned value into a signed value
- *Signed2Unsigned* for converting a signed value to an unsigned value
- *Real2Byte* for converting a floating point value into its binary representation, and
- *Byte2Real* for converting the binary representation of a floating point number into an analog value.

The connectors of the components for bytes, words and double words are integer connectors. Raw values are integer values too.

Bit2Byte – Converting bits into bytes

Symbol



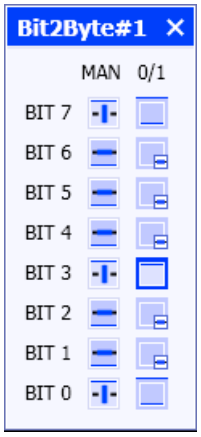
Function

The *Bit2Byte* component type converts the binary input values b_i, $i = 0, \ldots, 7$, into a byte value B at the integer output according to

$$B = \sum_{i=0}^{7} b_i \cdot 2^i$$

In the conversion, b_0 therefore represents the least significant bit and b_7 the most significant bit.

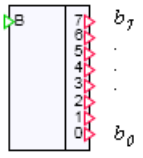
The individual bits b_i may be set individually while the simulation is running in the component operating window. To set a bit, open the component operating window. Then select the bits that you wish to set manually by pressing the button to the left of those bits. You can then set and reset each bit by pressing the button on the right, which means toggle between zero and one. In the figure below, bits b_0, b_3 and b_7 are selected to be set and bit b_3 is set.



Unset bits are identified by a light blue border, while set bits have a dark blue border.

Byte2Bit – Converting bytes into bits

Symbol

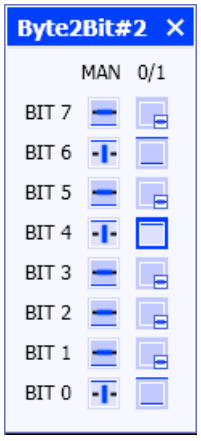


Function

The *Byte2Bit* component type converts a byte value at the integer input *B* into binary values b_i , i = 0, ... ,7 at the outputs. In the conversion, b_0 will be the least significant bit and b_7 the most significant bit of the byte value.

The input is limited to the range of values of one byte, which means to the range from zero to 255. If the input value is not within this range during the simulation, the message "*Byte2Bit: input not a valid byte value*" (message category *ERROR*) is generated.

The individual output bits b_i may be set individually while the simulation is running in the component operating window. To set a bit, open the component operating window. Then select the bits that you wish to set manually by pressing the button to the left of those bits. You can then set and reset each bit by pressing the button on the right, which means toggle between zero and one. In the figure below, bits b_0, b_4 and b_6 are selected to be set and bit b_4 is set:



Byte2Word – Converting bytes into words

Symbol



Function

The *Byte2Word* component type combines two byte values B_1, B_0 at the integer inputs to form a word at the integer output *W*. B_1 is the most significant byte and B_0 is the least significant byte.

The values at the input are limited to the range of values of one byte, which means to the range from zero to 255. If an input value is not within this range during the simulation, the message "*Byte2Word: input not a valid byte value*" (message category *ERROR*) is generated.

Word2Byte – Converting words into bytes

Symbol



Function

The *Word2Byte* component type breaks down a word at the integer input *W* into two byte values B_1, B_0 at the integer outputs. B_1 is the most significant byte and B_0 is the least significant byte.

The value at the input is limited to the range of values of one word, which means to the range from zero to 2^{16}–1. If the input value is not within this range during the simulation, the message "*Word2Byte: input not a valid word value*" (message category *ERROR*) is generated.

Byte2DWord – Converting bytes into double words

Symbol



Function

The *Byte2DWord* component type combines four byte values B_i, $i = 0, \ldots, 3$, at the integer inputs to form a double word in the order $B_3 - B_2 - B_1 - B_0$ at the integer output *DW*. B_3 is therefore the most significant byte and B_0 is the least significant byte.

The values at the input are limited to the range of values of one byte, which means to the range from zero to 255. If an input value is not within this range during the simulation, the message "*Byte2DWord: input not a valid byte value*" (message category *ERROR*) is generated.

DWord2Byte – Converting double words into bytes

Symbol



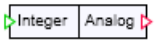
Function

The *DWord2Byte* component type converts a double word at the integer input *DW* into four binary values B_i, $i = 0, \ldots, 3$ at the integer outputs. B_3 is the most significant byte and B_0 is the least significant byte.

The value at the input is limited to the range of values of one double word, which means to the range from zero to $2^{32}–1$. If the input value is not within this range during the simulation, the message "*DWord2Byte: input not a valid double word value*" (message category *ERROR*) is generated.

Integer2Analog – Converting from integer to analog

Symbol

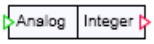


Function

The *Integer2Analog* component type converts an integer value at the input into an analog value at the output.

Analog2Integer – Converting from analog to integer

Symbol

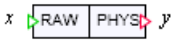


Function

The *Analog2Integer* component type converts an analog value at the input into an integer value at the output. The integer value is rounded.

Raw2Phys – Converting from raw to physical

Symbol



Function

The component type *Raw2Phys* converts an integer *x* at the input to an analog output value *y* in accordance with simple linear transformation:

$$\frac{y - y_L}{x - x_L} = \frac{y_U - y_L}{x_U - x_L}$$

The input value *x* may be a so-called raw value from an automation system, for example. The output value *y* is then the value mapped within a defined physical range of values.

Parameter

Parameter name	Description	Unit	Default value
Phys_Lower_Limit	Low limit for the transformation interval of the integer value	–	0.0
Phys_Upper_Limit	High limit for the transformation interval of the integer value	–	100.0
Raw_Lower_Limit	Low limit for the transformation interval of the analog value	–	–27648
Raw_Upper_Limit	High limit for the transformation interval of the analog value	–	27648

You can change the parameters during the course of the simulation.

Phys2Raw – Converting from physical to raw

Symbol



Function

The component type *Phys2Raw* converts an analog value *x* at the input to an integer output value *y* in accordance with simple linear transformation:

$$\frac{x - x_L}{y - y_L} = \frac{x_U - x_L}{y_U - y_L}$$

The input value *x* is thus transformed as a measurement of a physical value into the raw value *y* for an automation system, for example.

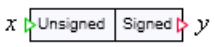
Parameter

Parameter name	Description	Unit	Default value
Phys_Lower_Limit	Low limit for the transformation interval of the analog value	–	0.0
Phys_Upper_Limit	High limit for the transformation interval of the analog value	–	100.0
Raw_Lower_Limit	Low limit for the transformation interval of the integer value	–	–27648
Raw_Upper_Limit	High limit for the transformation interval of the integer value	–	27648

You can change the parameters during the course of the simulation.

Unsigned2Signed – Converting from unsigned to signed

Symbol



Function

The *Unsigned2Signed* component type converts an unsigned integer value *x* at the input to a signed value *y* at the output.

The parameter *Width* determines which data width is assigned to the input value: *1 byte*, *2 bytes* or *4 bytes*. The output value is limited in accordance with the set data width. The conversion is done as follows:

Data width: 1 byte

$$y = \begin{cases} -1, & \text{if} \quad x > 2^8 - 1 \\ 0, & \text{if} \quad x < 0 \\ x - 2^8, & \text{if} \quad x > 2^7 - 1 \\ x, & \text{otherwise} \end{cases}$$

Data width: 2 bytes

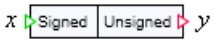
$$y = \begin{cases} -1, & \text{if} \quad x > 2^{16} - 1 \\ 0, & \text{if} \quad x < 0 \\ x - 2^{16}, & \text{if} \quad x > 2^{15} - 1 \\ x, & \text{otherwise} \end{cases}$$

Data width: 4 bytes

$$y = \begin{cases} -1, & \text{if} \quad x > 2^{32} - 1 \\ 0, & \text{if} \quad x < 0 \\ x - 2^{32}, & \text{if} \quad x > 2^{31} - 1 \\ x, & \text{otherwise} \end{cases}$$

Signed2Unsigned – Converting from signed to unsigned

Symbol



Function

The *Signed2Unsigned* component type converts a signed integer value *x* at the input to an unsigned value *y* at the output.

The parameter *Width* determines which data width is assigned to the output value: *1 byte*, *2 bytes* or *4 bytes*. The output value is limited in accordance with the set data width. The conversion is done as follows:

Data width: 1 byte

$$y = \begin{cases} -2^7, & \text{if} \quad x < -2^7 \\ 2^7 - 1, & \text{if} \quad x > 2^7 - 1 \\ x + 2^8, & \text{if} \quad x < 0 \\ x, & \text{otherwise} \end{cases}$$

Data width: 2 bytes

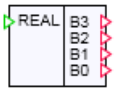
$$y = \begin{cases} -2^{15}, & \text{if} \quad x < -2^{15} \\ 2^{15} - 1, & \text{if} \quad x > 2^{15} - 1 \\ x + 2^{16}, & \text{if} \quad x < 0 \\ x, & \text{otherwise} \end{cases}$$

Data width: 4 bytes

$$y = \begin{cases} -2^{31}, & \text{if} \quad x < -2^{31} \\ 2^{31} - 1, & \text{if} \quad x > 2^{31} - 1 \\ x + 2^{32}, & \text{if} \quad x < 0 \\ x, & \text{otherwise} \end{cases}$$

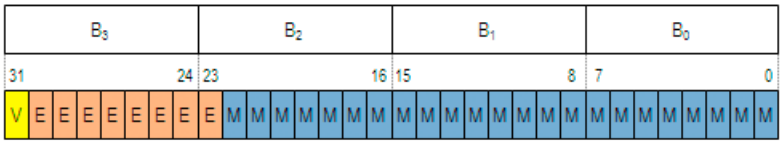
Real2Byte – Converting from real to byte

Symbol



Function

The component type *Real2Byte* converts the values of an analog signal at the *REAL* input to the binary representation of an single-precision floating-point number ("single" type) in accordance with IEEE 754. The converted floating-point number is mapped to the four byte values B_i, i = 0, ... ,3, at the integer outputs as illustrated in the figure below.



V	Sign
E	Exponent
M	Mantissa

Note

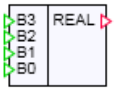
Mapping the analog signal (type double) to the single precision number format (type single) limits the precision and the value range.

This conversion cannot be used to convert a floating-point number to a whole number (integer). The *Analog2Integer* component type should be used for this conversion.

If you want to transfer analog signals to a SIMATIC controller, you normally set the data type of the corresponding signal in the coupling to *REAL*. The analog signal is then converted automatically. A component of the *Real2Byte* type is necessary, however, if you want to transfer analog signals directly to bit memories or data blocks of a SIMATIC controller, for example.

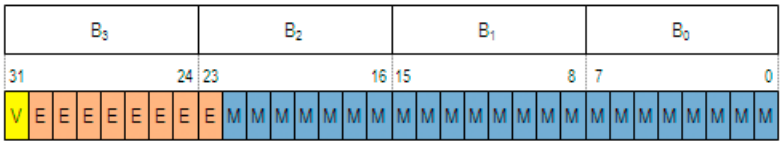
Byte2Real – Converting from byte to real

Symbol



Function

The *Byte2Real* component type converts the binary representation of a floating-point number of type "single" as defined in IEEE 754 at the integer inputs B_i, $i = 0, \ldots, 3$ to the value of an analog signal at the *REAL* output as shown in the figure below.



V	Sign
E	Exponent
M	Mantissa

Note

This conversion cannot be used to convert a a whole number (integer) into a floating-point number. The *Analog2Integer* component type should be used for this conversion.

If you want to receive floating-point numbers from a SIMATIC controller, you normally set the data type of the corresponding signal in the coupling to *REAL*. The conversion to an analog signal then takes place automatically. A component of the *Byte2Real* type is necessary, however, if you want to transfer floating-point numbers directly from bit memories or data blocks of a SIMATIC controller, for example.

8.1.3.7 General components in the Misc directory

Connectors

Three connector component types – referred to simply as connectors – are available:

- The analog connector component type *AConnector*,
- the binary connector component type *BConnector* and
- the integer connector component type *IConnector*.

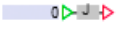
Connector components



Analog connector



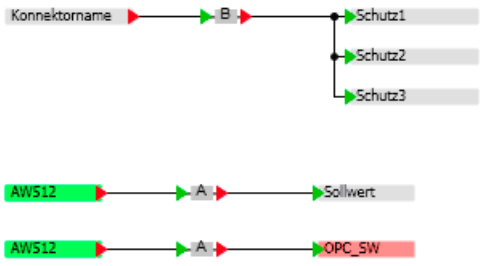
Binary connector



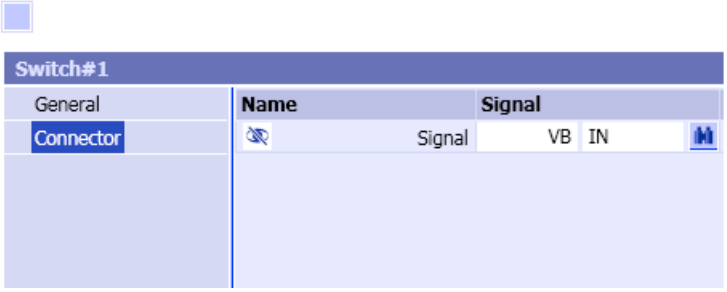
Integer connector

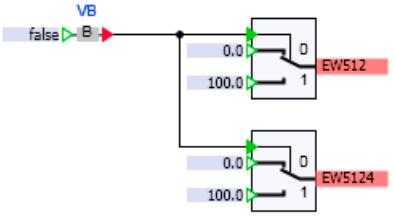
A connector component transfers the input value to its output without change or delay. They are used, for example, in the following cases:

- Connectors (I/O connectors or global connectors) cannot be connected directly to each other. As shown in the example in the figure below, connectors can be connected with the aid of connector components.

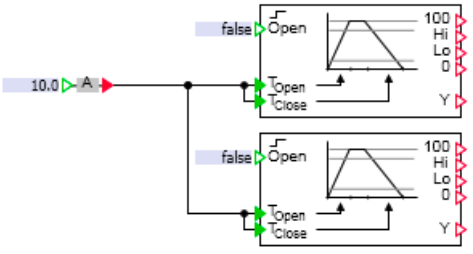


- An input element, for example a converter, is linked directly to the signal to be set. With the aid of a connector, the input element can influence several components directly. As shown in the example in the figure below, the input signal *IN* of the connector *VB* is linked to the input element.





- Component inputs can be directly assigned a default value. With the aid of a connector, several inputs can be assigned a single default value. As shown in the example in the figure below, a connector is connected to the inputs of the components to be set. The value to be set is then set at the input of the connector.



Auxiliary components for the creation of macros

Three so-called auxiliary component types are provided for macros:

- The analog constant component type *AConst*,
- The binary constant component type *BConst* and
- The integer constant component type *IConst*

Constant component types for macros

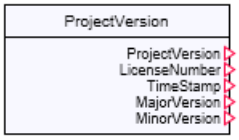
	Analog constant
	Binary constant
	Integer constant

Parameters of a macro component can only be built from parameters of the components that are used within the macro component. The constant component types thus also provide a parameter for setting inputs of a component as a macro parameter.

You can find additional information on this in the section: Defining parameters of macro components (Page 276)

ProjectVersion – Project version

Symbol



Function

The following integer values are available at the outputs of a component of this type:

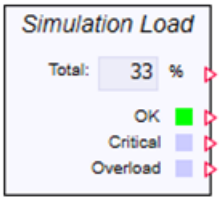
- **ProjectVersion**
  The complete version number coded in numerical form.
- **LicenseNumber**
  The license number coded in numerical form.
- **TimeStamp**
  The time stamp
- **MajorVersion**
  The major version
- **MinorVersion**
  The minor version

Operating window

The individual version numbers are displayed in readable form in the component operating window.

SimulationLoad

Symbol



Function

The *SimulationLoad* component type shows the current simulation load. You can find information about the simulation load in the section: Display of the simulation load (Page 35).

The symbol shows the total load in % under "Total". All values in this component are not limited and not smoothed. The values may therefore fluctuate more than the value in the symbol for the simulation in progress.

The three binary displays *OK*, *Critical* and *Overload* return a rough assessment of the load value as follows:

Load < 40%	OK
40% < Load < 90%	Critical
Load > 90%	Overload

The time slice containing the component for the load evaluation only has an impact on how often the values are updated. However, SIMIT always evaluates the values of all eight possible time slices. The worst value since the last update is used.

Operating window

This component shows the load values for all eight time slices in the operating window. The following causes are distinguished:

- Components (Components)
- Solution procedures (Solvers)
- Couplings (Gateways)

SIMIT distinguishes between *Time* and *CPU-Ticks*.

Time

Time is the percentage ratio of actual cycle time to configured cycle time.

The model with the worst time value is closest to the overload limit. However, this does not necessarily mean that there is too much calculation in this time slice.

The computational load is distributed across multiple available cores. Usually, a computing core must work through more than one time slice. Time slices with shorter cycle times have a higher priority. This means that the calculation of a slower time slice may be interrupted to permit the calculation of a faster time slice. This is important to calculate all cycles within the configured cycle time, if possible.

The result can be that the calculation in a slow time slice is constantly being interrupted by the calculation of a faster time slice, thereby creating a load problem. The faster time slice, in contrast, is calculated with a higher priority and can therefore have a lower load.

CPU-Ticks

To better identify the real cause of a high simulation load, *CPU-Ticks* are also available. They do not indicate the time that was needed for the calculation, but rather a measure for the time spent on the calculation operations. They do not take into account wait times due to interruptions by other time slices. As the measured absolute value of CPU-Ticks is not very meaningful, it is scaled:

$$ModelTicks_i' = ModelTicks_i \frac{MAX_{n=1..8}\,(Modeltime_n + Solvertime_n)}{MAX_{n=1..8}\,(Modelticks_n + Solverticks_n)}$$

$$SolverTicks_i' = SolverTicks_i \frac{MAX_{n=1..8}\,(Modeltime_n + Solvertime_n)}{MAX_{n=1..8}\,(Modelticks_n + Solverticks_n)}$$

$ModelTicks_n'$: Displayed value for the computational load of the component code in time slice *n*

$SolverTicks_n'$: Displayed value for the computational load of the solution procedure in time slice *n*

$ModelTicks_n$: Measured value for the computational operations of the component code in time slice *n*

$SolverTicks_n$: Measured value for the computational operations of the solution procedure in time slice *n*

$ModelTime_n$: Ratio of the time required for the calculation of the component code to the configured cycle time in time slice *n*

$SolverTime_n$: Ratio of the time required for the calculation of the solution procedure to the configured cycle time in time slice *n*

The ratio of tick values to each other shows where most arithmetic operations are performed in a given time. Relief should be provided in the time slice with the highest values, for example by relocating parts of the simulation model to a slower time slice or setting the entire critical time slice to a slower speed.

Simulation load by couplings

When calculating the tick values, it is assumed that the components and solution procedures only perform calculations and do not perform any file or console tasks. Although these commands require time, they do not fully utilize the calculation cores and therefore lead to misleading information.

Therefore, the couplings are not included in the count of arithmetic instructions. If a high simulation load is caused by a coupling, there are two reasons for this:

- The coupling is being interrupted too often by higher priority calculations.
- The coupling itself demands a great deal of time.

In the second case, reducing the computational work in the area of components or solution procedure is not very helpful. In general, the coupling must be assigned to a slower time slice, the signal range of the coupling must be reduced, or a part of the signals must be relocated to a second coupling with a slower cycle time.

Link

The statistical information about the load history is also visualized with a bar graph on the link of the component.

Output signals

The *SimulationLoad* component has output signals for operating controls in the operating window. These signals cannot be interconnected to the component symbol, but can be monitored in the property view.

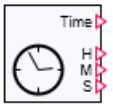
Output signal	Meaning
$LoadModel_nTicks$	Normalized computational work for the component code in time slice n
$LoadModel_nTime$	Percentage ratio of the time required for the calculation of the component code to the configured cycle time in time slice n
$LoadSolver_nTicks$	Normalized computational work for the solution procedure in time slice n
$LoadSolver_nTime$	Percentage ratio of the time required for the calculation of the solution procedure to the configured cycle time in time slice n
$LoadGateway_nTime$	Percentage ratio of the time required for the calculation of the coupling to the configured cycle time in time slice n

Extended operating window

In the extended operating window, the load information is further broken down according to the subdivision option of time slices A-H.

SimulationTime – Simulation time

Symbol



Function

The component type *SimulationTime* returns the current simulation time at its four outputs. The output takes place at the *Time* output as an integer value in milliseconds. The simulation time is available at the *H*, *M* and *S* integer outputs, broken down into hours (*H*), minutes (*M*) and seconds (*S*).

The simulation time is the time that is counted in the simulation cycle when simulation is performed. The simulation time is set to zero when a simulation is initialized or started. The simulation time and real time run synchronously if the simulation is running in real-time mode. In slow-motion simulation mode, the simulation time is slower than real time; in quick-motion simulation mode, the simulation time is faster than real time. If the simulation is paused, the simulation time is stopped as well. When the simulation is resumed, the simulation time starts running again as well.

Note

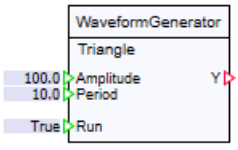
Resetting the simulation time can lead to undesired results if simulation mechanisms are directly dependent on the simulation time, such as trend display or isochronous mode.

Operating window

The simulation time is displayed in the operating window. You can reset the simulation time with *Reset*.

WaveformGenerator – Function generator

Symbol



Function

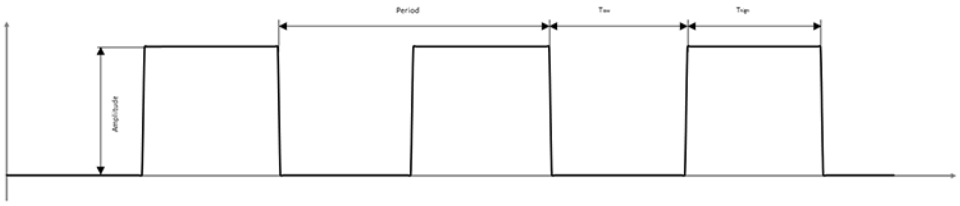
The *WaveformGenerator* component type generates periodic signals at output *Y*. Input *Run* must be set to True for output *Y* to be calculated dynamically. If *Run* has the value False, the Y value last calculated remains.

The *Period* input is specified in seconds; *Amplitude* is an absolute numerical value. The significance of these two inputs differs depending on the *Waveform* setting.

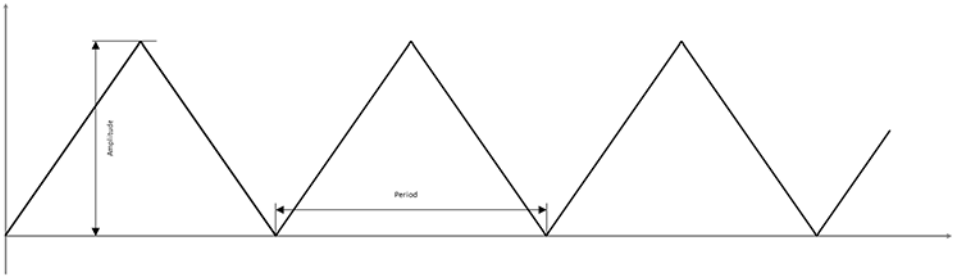
The *SquareRate* input is only active with the *Square* setting and sets the percentage ratio of *Tlow* to *Thigh*.

$$SquareRate\,[\%] = \frac{\mathrm{Tlow}}{Thigh}$$

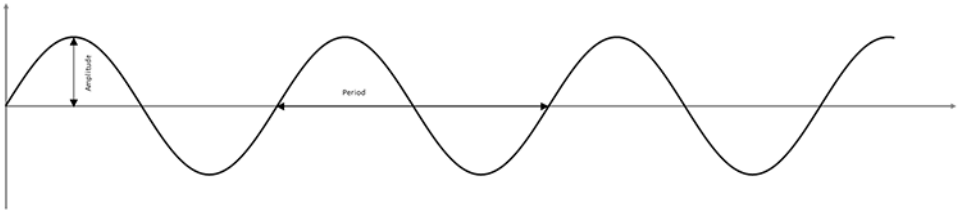
Square



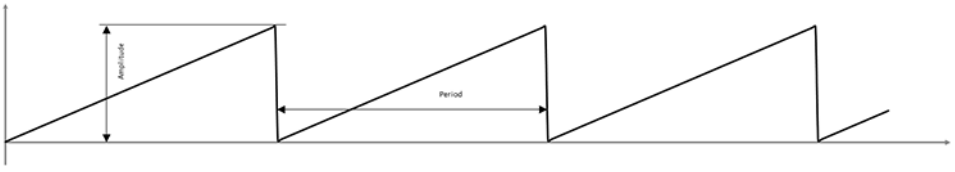
Triangle



Sine



Sawtooth

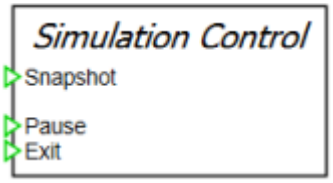


Parameter

Parameter name	Description	Unit	Default value
Waveform	Specifies the type of function.	–	*Triangle*

SimulationControl

Symbol



Function

The *SimulationControl* component allows creation of snapshots from the simulation model and pausing or exiting of the simulation.

It is also possible to create a snapshot and to pause or exit the simulation directly afterwards.

Operating window

A snapshot ("Snapshot") can be created via the operating window of the component.

The simulation can be paused ("Pause") or exited ("Exit").

A snapshot can be additionally created prior to pausing or exiting ("Snapshot & Pause" or "Snapshot & Exit").

Connectors

Input	Description
Snapshot	A snapshot is created at a positive edge.
Pause	The simulation is paused but not exited at a positive edge. If the value "true" is simultaneously active at the "Snapshot" input, a snapshot will be created before pausing.
Exit	The simulation is exited at a positive edge. If the value "true" is simultaneously active at the "Snapshot" input, a snapshot will be created before exiting.

8.1.4 Drive components

8.1.4.1 Drives as component types

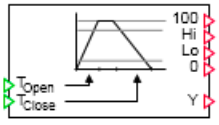
In the directory *DRIVES* of the Basic Library you will find component types for simulation of drives. These component types form the Drive Library. They can be used to simulate general drives for valves and pumps. The types

- *DriveP1* and *DriveP2* are designed for simulating pump drives, while the types
- *DriveV1* to *DriveV4* are intended for simulating valve drives.

The subdirectory *PROFIdrives* contains components for the simulation of variable-speed drives corresponding to the PROFIdrive profile. Components for the simulation of SIMOCODE motor management and control devices can be found in the subdirectory *SIMOCODEpro*.

8.1.4.2 Valve drives

Four component types are available (*DriveV1*, *DriveV2*, *DriveV3* and *DriveV4*) to simulate valve drives. All four of these component types share the outputs and some of the inputs as illustrated in the figure below.



The two analog inputs *TOpen* and *TClose* indicate the opening and closing times of the drive, which means the time that the drive takes to fully open or close. If one of the two input values is negative during simulation, the message "*DriveVx: closing or opening time invalid value*" (message category *ERROR*) is generated.

The current position value of the drive is output at analog output *Y* as a percentage, which means in the range from zero to one hundred:

$0 \leq Y \leq 100$.

The value zero corresponds to the fully closed valve, while the value one hundred represents the fully open valve.

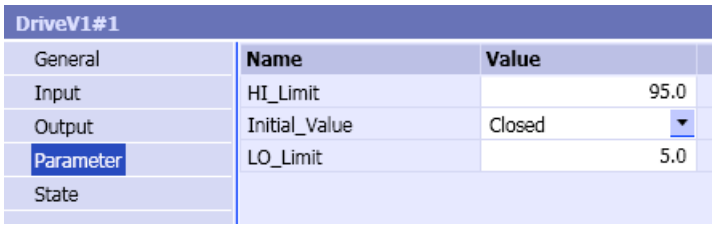
The four binary outputs *100*, *Hi*, *Lo* and *0* may be interpreted as limit switches for the drive:

- *100* and *0* as limit switches "Valve fully opened" or "Valve fully closed",
- *Hi* and *Lo* as pre-limit switches for "Valve open" or "Valve closed".

The limit switches are permanently set to the position values zero (for the fully closed valve) and one hundred (for the fully open valve). Binary outputs *100* or *0* are therefore set to one when the valve is fully closed or open, which means when the position *Y* of the component has the value zero or one hundred.

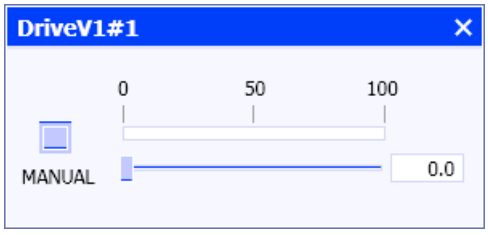
The *Hi* and *Lo* pre-limit switches can be set using the *HI_Limit* or *LO_Limit* parameters. Their values are between 0 and 100. Position values of five (*LO_Limit*) and 95 (*HI_Limit*) are set by default.

If one of the two parameter values is negative during the simulation, the message "*DriveVx: high and low parameters do not match*" (message category *ERROR*) is generated.



The *Initial_Value* parameter is used to set the valve drive to the starting position of *Closed* or *Open*. The default setting for *Initial_Value* is *Closed*.

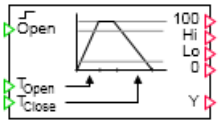
The current position value of the drive is visualized as a bar graph display in the component operating window. The button on the left labelled *MANUAL* is used to change the position value to the desired setting by moving the slider in the operating window.



The position value is then no longer derived from the component inputs, but rather tracks the value set using the slider. The set opening and closing times remain active. The four binary outputs *100*, *Hi*, *Lo* and *0* likewise all remain effective. They are also set as described above if the position value is to be tracked.

DriveV1 – Type 1 valve drive

Symbol

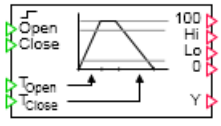


Function

The *DriveV1* component type simulates a drive unit that sets the position value *Y* at the output in relation to a binary value at the input *Open*. If the binary input is equal to zero, the position value *Y* is continuously tracked to zero with the closing time T_{Close}. If the binary input is equal to one, the position value is continuously tracked to one hundred with the opening time T_{Open}. The position value thus only has the two steady states of zero and one hundred, which means the states "Valve open" and "Valve closed" in relation to the valve function. The drive or the valve is thus opened or closed whenever the binary value at the input changes.

DriveV2 – Type 2 valve drive

Symbol



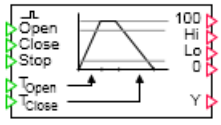
Function

The *DriveV2* component type simulates a drive unit that sets the position value *Y* at the output in relation to two binary values at the *Open* and *Close* inputs. If the binary value at input *Open* is equal to one, the position value *Y* is continuously tracked to one hundred with the opening time T_{Open}. If the binary value at the input *Close* is equal to one, the position value is continuously tracked to zero with the closing time T_{Close}.

If both input values are set to one or zero at the same time, the position value remains unchanged. The position value thus only changes if the binary value at either the *Open* input or the *Close* input is set to one.

DriveV3 – Type 3 valve drive

Symbol

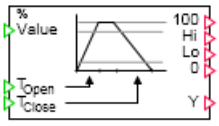


Function

The *DriveV3* component type simulates a drive unit that sets the position value *Y* at the output in relation to three binary values at the *Open*, *Close* and *Stop* inputs. If the binary value at the *Open* input changes from zero to one, the position value *Y* is continuously tracked to one hundred with the opening time T_{Open}. The position value is tracked continuously to zero with the closing time T_{Close} if the binary value at the *Close* input changes from zero to one. If the binary value at the *Stop* input changes from zero to one, the position value remains unchanged. Opening, closing and stopping of the drive is thus initiated via the edge of the corresponding binary input signal (signal change from zero to one).

DriveV4 – Type 4 valve drive

Symbol

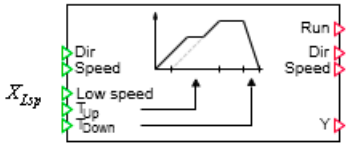


Function

The *DriveV4* component type simulates a drive unit that sets the position value *Y* at the output that continuously tracks the analog value at the *Value* input. The tracking occurs in the direction of rising position values with the opening time T_{Open} and falling position values with the closing time T_{Close}. The position value is always limited to values from zero to one hundred, even if the input value is outside this interval.

8.1.4.3 Pump drives and fan drives

The two component types *DriveP1* and *DriveP2* may be used in the simulation as drives for pumps, fans or similar units. The outputs and some of the inputs common to these two component types are illustrated in the figure below:



The run-up and run-down speeds of the drive are set at the two analog inputs T_{Up} and T_{Down}. T_{Up} is the time it takes the drive to run up from standstill to the nominal speed in seconds, while T_{Down} is the time in seconds it takes to run the drive down from nominal speed to standstill. Both times have a default value of one second. If one of the two input values is negative during simulation, the message "*DriveVx: run-up or run-down time invalid value*" (message category *ERROR*) is generated.

The current speed value of the drive is output at analog output *Y* as a percentage, which means in the range from zero to one hundred:

$0 \leq Y \leq 100$.

The value zero corresponds to the stopped drive, while the value one hundred represents the drive at nominal speed.

The direction of rotation of the drive can be set using binary input *Dir*. When this input is set to zero, the direction of rotation is positive. If the input is set to one, the direction of rotation is negative. This input can therefore be used to specify whether the drive turns clockwise or counterclockwise, for example. If it turns in the negative direction, the speed is output as a negative value at output *Y*:

$-100 \leq Y \leq 0$.

Positive values for *Y* thus indicate speeds in the positive direction, while negative values indicate speeds in the negative direction. The change in direction of rotation only takes effect if the drive is at a standstill.

The binary input *Speed* is used to toggle the speed between nominal speed (full speed) and partial speed (low speed). If *Speed* is one, the nominal speed was selected; otherwise the partial speed was selected. The default setting for the *Speed* binary input is one. The partial speed is specified as a numerical value (percentage) at the analog input x_{LSp}:

$0 \leq x_{LSp} \leq 100$.

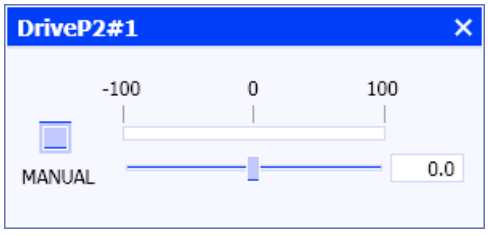
The default setting for the partial speed is 50, which means half the nominal speed. If the partial speed is not within the defined range, the message "*DrivePx: low speed invalid value*" (message category *ERROR*) is generated.

The binary output *Run* is only set to one when the drive has reached the preset speed value in the positive or negative direction of rotation, which means only if the absolute value at the analog output *Y* is equal to one hundred (nominal speed) or is equal to the partial speed set at the input x_{LSp}.

The feedback signal for the selected speed is available at the binary output *Speed*. The binary output is only set to one if the drive has reached its nominal speed in the positive or negative direction of rotation.

The value at the binary output *Dir* takes the form of a feedback signal for the current direction of rotation of the drive. The binary output is zero if the drive is turning in the positive direction; the value is one if the drive is turning in the negative direction.

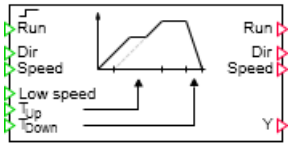
The current speed value of the drive is visualized as a bar graph display in the component operating window. The button on the left labelled *MANUAL* is used to change the speed value to the desired setting by moving the slider in the operating window.



The speed value is then no longer derived from the component inputs, but rather tracks the value set using the slider. The specified start-up and run-down times remain effective. The binary outputs *Run*, *Speed* and *Dir* likewise remain active. They are also set as described above if the speed value is to be tracked.

DriveP1 – Type 1 pump drive

Symbol

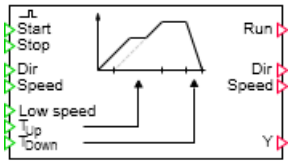


Function

The *DriveP1* component type simulates a drive unit that is switched on and off via the binary value at the *Run* input. The drive is switched on while the input value is set to one. If the input value is set to zero, the drive is switched off.

DriveP2 – Type 2 pump drive

Symbol



Function

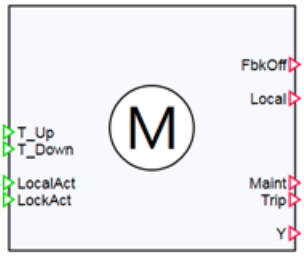
The *DriveP2* component type simulates a drive unit that is switched on and off via the two binary inputs *Start* and *Stop*. If the binary value *Start* changes from zero to one, then the drive is switched on. If the binary value *Stop* changes from zero to one, the drive is switched off. Activation and deactivation of the drive is initiated with the edge of the corresponding binary input signal (signal change from zero to one).

8.1.4.4 Motor

Common features of motor components

Shared I/O of component types for pump drives

The component types *Motor, ReversibleMotor* and *TwoSpeedMotor* can be used in the simulation as drives for pumps, fans or similar units. Some of the inputs and outputs are the same for all components as shown in the figure below.



Analog inputs

The start-up and run-down times of the drive are specified at the analog inputs *T_UP* and *T_DOWN*.

T_UP is the time it takes the drive to run from standstill to the nominal speed in seconds; *T_DOWN* is the time in seconds it takes to run the drive down from nominal speed to standstill. The default time is 1 s.

If one of the two input values is negative during the simulation, the message "*Motor: run-up or run-down time invalid value*" (message category *ERROR*) is generated.

Binary inputs

Binary input *LocalAct* switches the drive to the local operating mode. In this mode, the drive can only be controlled over the component operating window. The drive can be locked with the binary input *LockAct* so that neither local operation nor operation using the component control inputs is possible. The drive is shut down and locked if the component receives a rising edge at the *LockAct* input during operation.

Binary outputs

The binary output *FbkOff* is True when the drive is at a standstill (Y = 0). The binary output *Local* can be activated from the drive operating window. If the output is connected accordingly, it can send local operations to the controller. The binary output *Maint* can be activated from the drive operating window. The binary output *Trip* can be activated from the drive operating window. If Trip is enabled, the drive shuts down and remains at a standstill (Y =0).

Analog output

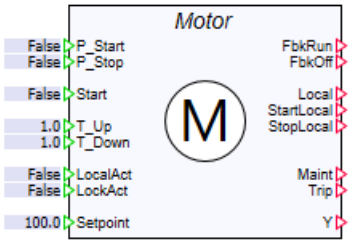
The current speed of the drive is output as a percentage at analog output Y, in other words in a range of 0 to 100:

$0 \leq Y \leq 100$ or with ReversibleMotor $-100 \leq Y \leq 100$.

A value of 0 corresponds to a drive at a standstill and a value of 100 to a drive at the nominal speed (or −100 with ReversibleMotor).

Motor

Symbol



Function

The component type *Motor* simulates a drive unit that is switched on and off by the binary value at the *Start* input. The drive is on for as long as the start input value is set to True. If the start input value is set to False, the drive is switched off.

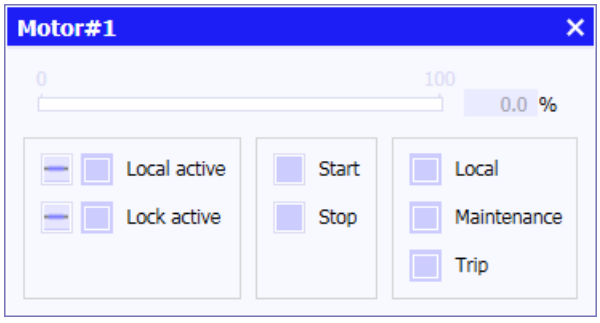
The component can also be switched on and off with the two binary inputs *P_Start* and *P_Stop* . If the binary signal at the *P_Start* input changes from False to True , the drive is switched on. If the binary signal at the *P_Stop* input changes from False to True, the drive is switched off.

The speed value of the drive can be specified as a percentage in the range from 0 to 100 via the analog input *Setpoint*. The default setting for the component is 100. If the speed value is not in this defined range, the message "*Motor: Setpoint invalid value*" (message category *ERROR*) is generated.

The binary output *FbkRun* is only set to True if the drive has reached the preselected speed in a positive direction: The value at analog output *Y* is identical to the value specified at the *Setpoint* input.

The binary outputs *StartLocal* and *StopLocal* only represent the duration of the key operation to set the *Start* or *Stop* key in the component operating window to True.

Operating window



The component operating window shows the current speed of the drive as a bar graph.

The left-hand column shows the two inputs *LocalAct* and *LockAct*. You can see the current state pending here.

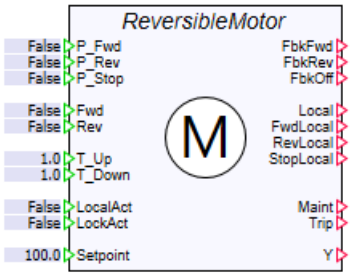
You can also separate both inputs from their sources and set the value manually.

If *LocalAct* (Local active) is enabled, the speed is no longer derived from the inputs of the component. Instead, it is formed from the local operating buttons in the central column *Start* and *Stop*. The specified start-up and run-down times remain effective. The switch from remote to local is smooth: If the drive was on before the switchover, it is initially also on immediately afterwards.

In the right-hand column is the operating switch for *Local*, maintenance (*Maint*) and *Trip*.

ReversibleMotor – Motor with 2 directions

Symbol



Function

The component type *ReversibleMotor* simulates a drive unit that is controlled by two binary values at the *Fwd* and *Rev* inputs. The drive is switched on in forwards mode as long as the input value *Fwd* is set to True. If the input value is set to False, the drive is switched off. The drive is switched on in reverse mode for as long as the input value*Rev* is set to True. If the input value is set to False, the drive is switched off.

The component can also be pulse-controlled over the three binary inputs *P_Fwd, P_Rev* and *P_Stop*. If the binary signal at the *P_Fwd* input changes from False to True, the drive is switched on in forwards mode. If the binary signal at the *P_Stop*input changes from False to True, the drive is switched off. If the binary signal at the *P_Rev* input changes from False to True , the drive is switched on in reverse mode.

The change in direction of rotation only takes effect if the drive is at a standstill.

The speed value of the drive can be specified as a percentage in the range from 0 to 100 via the analog input *Setpoint*. The default setting for the component is 100. If the speed value is not in this defined range, the message "*Motor: Setpoint invalid value*" (message category*ERROR*) is generated.

The binary output *FbkFwd* is only set to True if the drive has reached the preselected speed in a positive direction (forwards mode): The value at analog output *Y* is identical to the value specified at the *Setpoint* input.

The binary output *FbkRev* is only set to True if the drive has reached the preselected speed in a negative direction (reverse mode): The value at analog output *Y* is the negative of the value specified at the *Setpoint* input.

The binary outputs *FwdLocalRevLocal* and *StopLocal* only represent the duration of the key operation to set the *Forward*, *Reverse* or *Stop* key in the component operating window to True.

Operating window



The component operating window shows the current speed of the drive as a bar graph.

The left-hand column shows the two inputs *LocalAct* and *LockAct*. You can see the current state pending here.

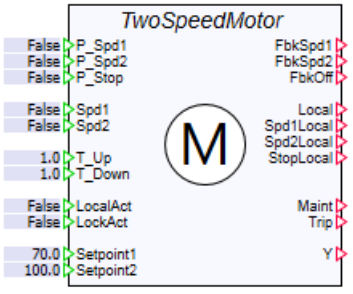
You can also separate both inputs from their sources and set the value manually.

If *LocalAct* (Local active) is enabled, the speed is no longer derived from the inputs of the component. Instead, it is formed from the local operating buttons in the central column, *Forward, Reverse* and *Stop*. The specified start-up and run-down times remain effective. The switch from remote to local is smooth: If the drive was on before the switchover, it is initially also on immediately afterwards.

In the right-hand column is the operating switch for *Local*, maintenance (*Maint*) and *Trip*.

TwoSpeedMotor – Motor with 2 speeds

Symbol



Function

The component type *TwoSpeedMotor* simulates a drive unit that is controlled by two binary values at the *Spd1* and *Spd2* inputs. The drive is switched on and run until *Setpoint1* as long as the input value *Spd1* is set to True. If the input value is set to False, the drive is switched off. The drive is switched on and run until *Setpoint2* as long as the input value *Spd2* is set to True. If the input value is set to False, the drive is switched off. If both the *Spd1* and *Spd2* input are pending at the same time, the drive is operated at *Spd1* .

The component can also be pulse-controlled over the three binary inputs *P_Spd1*, *P_Spd2* and *P_Stop*. If the binary signal at the *P_Spd1* input changes from False to True, the drive is switched on at the reduced percentage speed specified at the *Setpoint1* input. If the binary signal at the *P_Stop*input changes from False to True, the drive is switched off. If the binary signal at the *P_Spd2* input changes from False to True, the drive is switched on at the full speed specified at the *Setpoint2* input (max. 100). You can toggle directly between *Spd1* and *Spd2* .

You can set the reduced drive speed as a percentage of between 0 and 100 at the analog input *Setpoint1*. The default setting for this input is 70. You can set the full drive speed as a percentage of between 0 and 100 at the analog input *Setpoint2*. The default setting for this input is 100. If the speed values are not in these defined ranges, the message "*Motor: Setpoint invalid value*" (message category*ERROR*) is generated. The setpoints can be configured separately.

The binary output *FbkSpd1* is only set to True if the drive has reached the preselected speed: The value at analog output *Y* is identical to the value specified at the *Setpoint1* input.

The binary output *FbkSpd2* is only set to True if the drive has reached the preselected speed: The value at analog output *Y* is identical to the value specified at the *Setpoint2* input.

The binary outputs*Spd1Local*, *Spd2Local* and *StopLocal* only represent the duration of the key operation to set the *Speed 1*, *Speed 2* or *Stop* keys in the component operating window to True.

Operating window



The component operating window shows the current speed of the drive as a bar graph.

The left-hand column shows the two inputs *LocalAct* and *LockAct*. You can see the current state pending here.

You can also separate both inputs from their sources and set the value manually.

If *LocalAct* (Local active) is enabled, the speed is no longer derived from the inputs of the component. Instead, it is formed from the local operating buttons in the central column, *Speed 1*, *Speed 2* and *Stop*. The specified start-up and run-down times remain effective. The switch from remote to local is smooth: If the drive was on before switchover with Speed 1, it is initially also on immediately switchover with Speed 1.

In the right-hand column is the operating switch for *Local*, maintenance (*Maint*) and *Trip*.

8.1.4.5 PROFIdrive devices

Many variable-speed drives are connected directly to the controller via PROFIBUS DP or PROFINET. To standardize the configuration of such drives, a profile called PROFIdrive was created by the PROFIBUS User Organization (PNO). This profile defines how a drive matching this profile behaves on the field bus with regard to some of its more important functions. The following power converters or frequency converters from the Siemens range of products meets the requirements of a PROFIdrive:

- SIMOREG DC Master
- SIMOVERT Masterdrive
- Micromaster
- Sinamics

A common feature of all PROFIdrive devices is that they are activated by control words (STW) and provide feedback in a status words (ZSW). In addition, the drives receive a speed setpoint from the controller and provide the actual speed and an actual position as feedback.

The functionality of a PROFIdrive can be divided roughly into the following blocks:

- State machine
- Ramp generator for the transferred setpoint speed
- Position actual value preparation
- Position control (not included in all PROFIdrive telegrams)
- Torque limiting (only for SIEMENS-specific telegrams)

The state machine defines the transitions from one status to another. A change in status is triggered by a particular bit in control word 1. The current status can be seen in status word 1. The purpose of the ramp generator of a PROFIdrive is to convert jumps in the speed setpoint into linear ramp increases with an adjustable gradient.

In addition to this basic function, PROFIdrive devices can of course offer considerably more functions and configuration options.

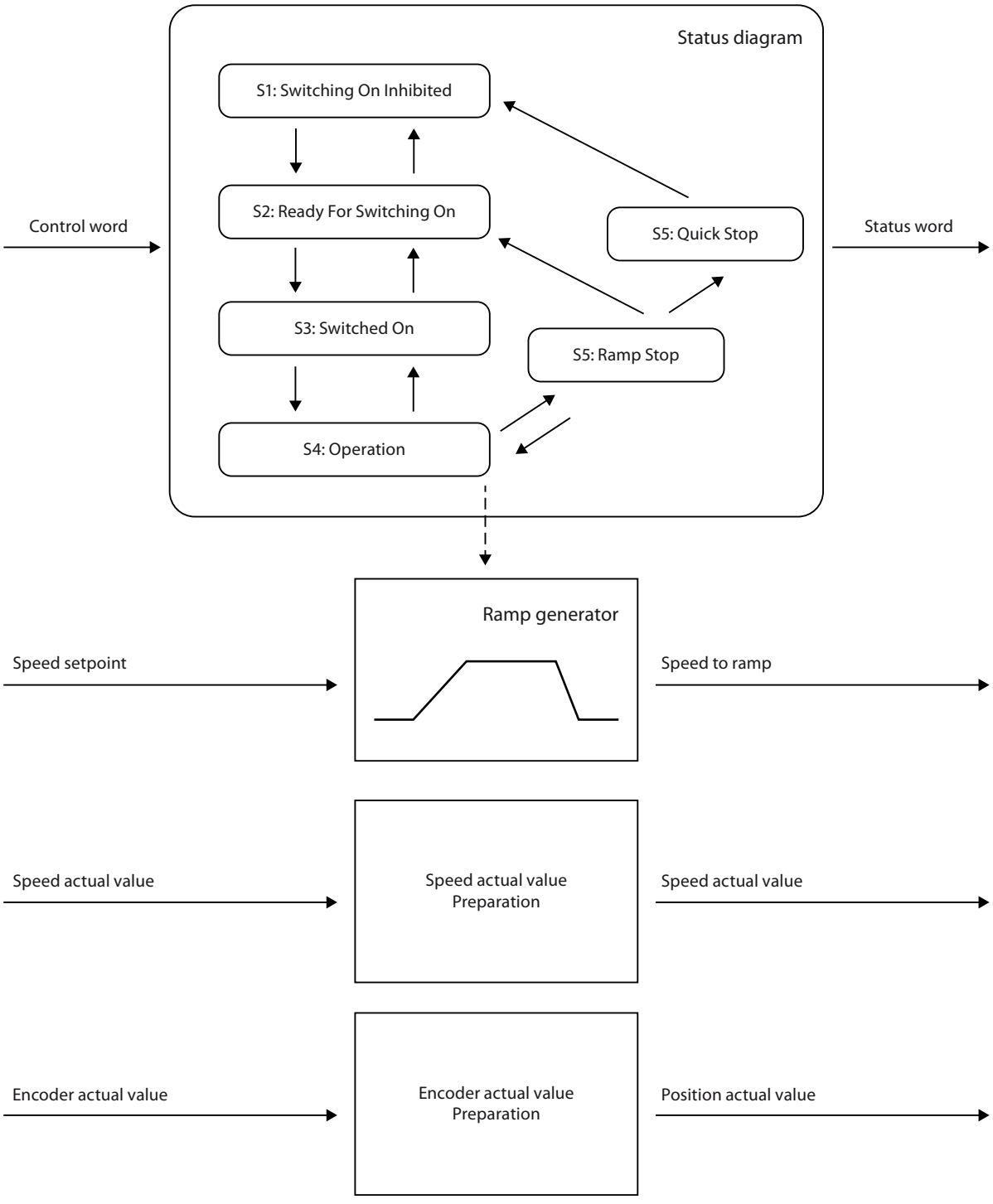
Using the PROFIdrive library to simulate PROFIdrive drives

All component types for the simulation of PROFIdrive devices can be found in the *PROFIdrive* directory of the Drives Library:

- The PROFIdrive header component type *PROFIdrive2*
- The sensor simulation for rotary and linear encoders
- The preparation of sensor data
- Position control in the drive
- The evaluation of the torque reduction for SIEMENS devices
- The support of safety functionality in the drive
- The adaptation of device-specific data with the controller
- The PROFIdrive device-specific component types *Sinamics* and *Universal*.

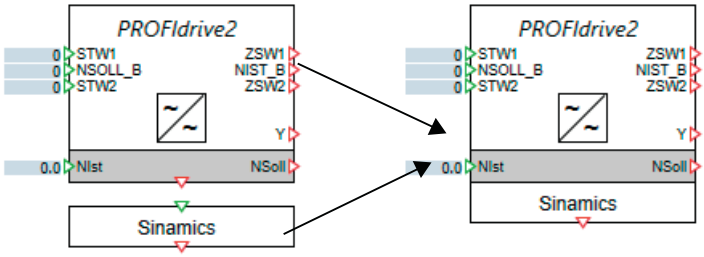
These component types make up the PROFIdrive library.

The state machine and ramp generator functions contained in all PROFIdrive devices are simulated in the *PROFIdrive* component type.



Some of the bits in the control and status words are not defined in the PROFIdrive profile. The five bits 11 to 15 in the control and status word 1 can be drive-specific and may therefore have different meanings according to the type of drive. For the SINAMICS and Universal AC

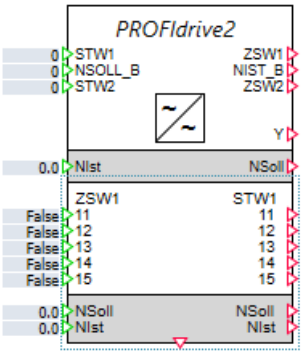
converters, these drive-specific functions are emulated in the corresponding device-specific components. These drive-specific components are simply attached to the *PROFIdrive* header component, as illustrated in the figure below.



The device-specific component types in the *PROFIdrive* library contain standardized emulations of different PROFIdrive drives. You can find additional information on this in section: Sinamics – SINAMICS frequency converter (Page 546).

The meaning of the drive-specific bits can be changed by configuring the converter systems and may therefore deviate from the standard configuration illustrated here. If this is the case, the device-specific component type is set to *Universal*.

You can find additional information on this in section: Universal – Extensions to the PROFIdrive basic function (Page 542). This component type provides you with bits 11 to 15 of control word 1 and status word 1 as component inputs and outputs, which can be connected as required to other components in the basic library.



In addition to the control and status word and the speed setpoint and actual speed values, PROFIdrive devices can exchange other information with the controller. These additional I/O signals are available in SIMIT through the SIMIT Unit coupling like standard I/O signals and can therefore also be used for specific modification of a drive simulation.

PROFIdrive drives – displayable PROFIdrive telegrams

The basic library contains multiple PROFIdrive components. They can be used to map the following PROFIdrive standard telegrams as well as SIEMENS-specific extended telegrams.

	PZD 1	PZD 2	PZD 3	PZD 4	PZD 5	PZD 6	PZD 7	PZD 8	PZD 9	PZD 10	PZD 11	PZD 12	PZD 13	PZD 14	PZD 15
TEL_1 Nominal	STW 1	NSOLL_A													
TEL_1 Actual	ZSW 1	NIST_A													
TEL_2 Nominal	STW 1	NSOLL_B		STW 2											
TEL_2 Actual	ZSW 1	NIST_B		ZSW 2											
TEL_3 Nominal	STW 1	NSOLL_B		STW 2	G1_STW										
TEL_3 Actual	ZSW 1	NIST_B		ZSW 2	G1_ZSW	G1_XIST 1		G1_XIST 2							
TEL_4 Nominal	STW 1	NSOLL_B		STW 2	G1_STW	G2_STW									
TEL_4 Actual	ZSW 1	NIST_B		ZSW 2	G1_ZSW	G1_XIST 1		G1_XIST 2		G2_ZSW	G2_XIST 1		G2_XIST 2		
TEL_5 Nominal	STW 1	NSOLL_B		STW 2	G1_STW	XERR		KPC							
TEL_5 Actual	ZSW 1	NIST_B		ZSW 2	G1_ZSW	G1_XIST 1		G1_XIST 2							
TEL_6 Nominal	STW 1	NSOLL_B		STW 2	G1_STW	G2_STW	XERR		KPC						
TEL_6 Actual	ZSW 1	NIST_B		ZSW 2	G1_ZSW	G1_XIST 1		G1_XIST 2		G2_ZSW	G2_XIST 1		G2_XIST 2		
TEL_102 Nominal	STW 1	NSOLL_B		STW 2	M_Red	G1_STW									
TEL_102 Actual	ZSW 1	NIST_B		ZSW 2	Meld_W	G1_ZSW	G1_XIST 1		G1_XIST 2						
TEL_103 Nominal	STW 1	NSOLL_B		STW 2	M_Red	G1_STW	G2_STW								
TEL_103 Actual	ZSW 1	NIST_B		ZSW 2	Meld_W	G1_ZSW	G1_XIST 1		G1_XIST 2		G2_ZSW	G2_XIST 1		G2_XIST 2	
TEL_105 Nominal	STW 1	NSOLL_B		STW 2	M_Red	G1_STW	XERR		KPC						
TEL_105 Actual	ZSW 1	NIST_B		ZSW 2	Meld_W	G1_ZSW	G1_XIST 1		G1_XIST 2						
TEL_106 Nominal	STW 1	NSOLL_B		STW 2	M_Red	G1_STW	G2_STW	XERR		KPC					
TEL_106 Actual	ZSW 1	NIST_B		ZSW 2	Meld_W	G1_ZSW	G1_XIST 1		G1_XIST 2		G2_ZSW	G2_XIST 1		G2_XIST 2	

The telegrams are modeled by lining up the individual components.

The following sections explain the "blocks", the available individual components, in more detail.

The "Safety enhancement" component plays a special role; it expands the standard PROFIdrive drives by safety functions.

In addition, a component is offered for the SIEMENS-specific telegram 370 to simulate an active infeed (see InFeed_Tel_370 (Page 544)).

DataAdaption – Functionality of data adaption of a PROFIdrive drive

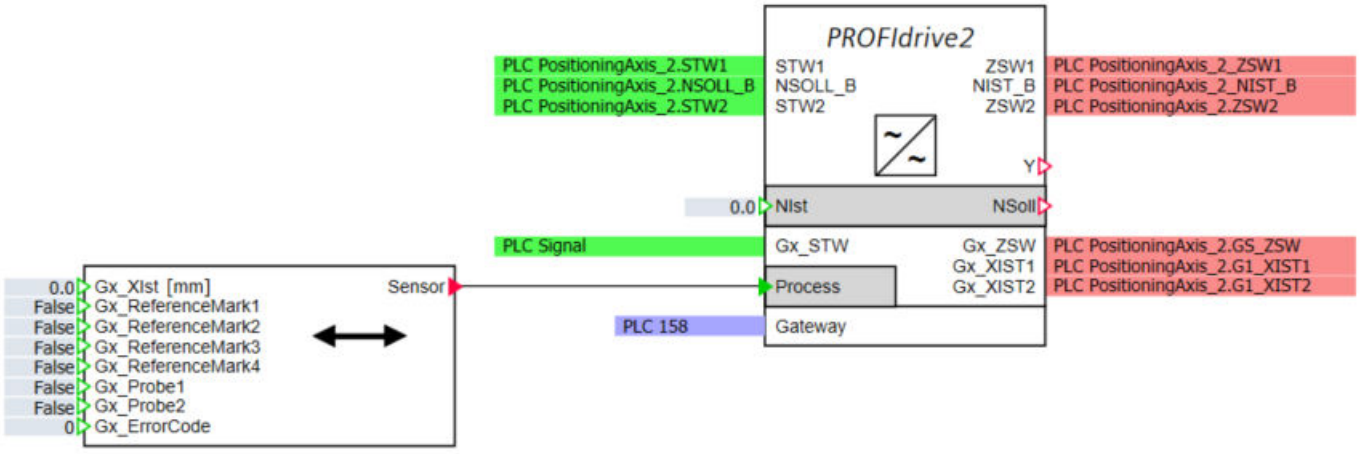
Symbol



Function

Technological objects in the S7-1500 have a functionality with which they synchronize their technological parameters with the technological parameters of the drive. This process is called data adaptation. The *DataAdaption* component type supplies the required information to the technology object using acyclic data record communication.

The DataAdaption component is always the last component to be added to a simulated drive.



Note

Currently, this component can only be used in conjunction with the PLCSIM Advanced coupling.

Connectors

Input	Description
Gateway	Connection to the configuration of the drive component via a unit connector
DriveIn	Structure for docking to upstream PROFIdrive components

Parameter

Parameters	Description
ReferenceSpeed	Here you enter the reference speed.

MaxSpeed	Here you enter the maximum physical speed of the drive.
G1_EncoderSystem	Enter the type of encoder system here. 0 = rotary encoder system 1 = linear encoder system
G1_SupportsAbsolutValues	Enter here whether encoder 1 is an True = absolute encoder False = incremental encoder
G1_DeterminableRevolutions	Specify here how many encoder revolutions can be determined for the absolute value.
G1_FineResolutionXist1	Enter the fine resolution of G1_Xist1 here.
G1_FineResolutionXist2	Enter the fine resolution of G1_Xist2 here.
G2_EncoderSystem	Enter the type of encoder system here. 0 = rotary encoder system 1 = linear encoder system
G2_SupportsAbsolutValues	Enter here whether encoder 2 is an True = absolute encoder False = incremental encoder
G2_DeterminableRevolutions	Specify here how many encoder revolutions can be determined for the absolute value.
G2_FineResolutionXist1	Enter the fine resolution of G2_Xist1 here.
G2_FineResolutionXist2	Enter the fine resolution of G2_Xist2 here.
DSC_Configured	Specify here whether you use a DynamicServoControl block in the drive block group.
Torque_Nominal	Here you enter the reference torque.
Torque_UpperLimit	Here you enter the top physical speed limit of your drive.
Torque_LowerLimit	Here you enter the low physical speed limit of your drive.
Torque_RedScale	Here you specify the percentage to which the torque reduction is to be normalized. The reference torque is typically 100%.
DeviceIdentificationCompany	Enter the manufacturer ID here. Siemens = 42
DeviceIdentificationDeviceType	Enter the device ID here. Sinamics S120 = 5011
Additional_Parameter	Free text for specifying additional parameters

Additional parameters can be specified as free text in the following format:

Parameter number [ParameterSubIndex] = format#value

For example: "180 [1] = u64#10" means Parameter 180, SubIndex 1 has the value 10 in unsigned64 data format.

The following data formats are defined:

- b – boolean
- i8 – integer8

- i16 – integer16
- i32 – integer32
- i64 – integer64
- u8 – unsigned8
- u16 – unsigned16
- u32 – unsigned32
- u64 – unsigned64
- f64 – floating point

The following table shows the parameters provided by the DataAdaption component to the technology object.

Parameter	Parameter text
51	Drive Data Set DDS effective
107	Drive object type / DO type
108	Drive objects function module
180	Number of Drive Data Sets
505	Selecting the system of units / Unit sys select
922	IF1 PROFIdrive PZD telegram selection
924 SubIndex 0	ZSW bit pulses enabled.Signal number
924 SubIndex 1	ZSW bit pulses enabled.Bit position
925	PROFIdrive clock synchronous sign-of-life tolerance
950 SubIndex 0	Number of fault situation
950 SubIndex 1	Number of fault messages in a fault situation
964 SubIndex 0	Device identification.Company
964 SubIndex 1	Device identification.Device type
964 SubIndex 2	Device identification.Firmware version
964 SubIndex 3	Device identification.Firmware date (year)
964 SubIndex 4	Device identification.Firmware date (day/month)
964 SubIndex 5	Device identification.Number of drive objects
964 SubIndex 6	Device identification.Firmware patch/hot fix
965	PROFIdrive profile number
975 SubIndex 1	Drive object identification.Drive object type
975 SubIndex 5	Drive object identification.drive object type class
978 SubIndex 0	List of drive objects
978 SubIndex 1	List of drive objects
978 SubIndex 2	List of drive objects
978 SubIndex 3	List of drive objects
978 SubIndex 4	List of drive objects
978 SubIndex 5	List of drive objects
978 SubIndex 6	List of drive objects
978 SubIndex 7	List of drive objects
978 SubIndex 8	List of drive objects

Parameter	Parameter text
978 SubIndex 9	List of drive objects
978 SubIndex 10	List of drive objects
978 SubIndex 11	List of drive objects
978 SubIndex 12	List of drive objects
978 SubIndex 13	List of drive objects
978 SubIndex 14	List of drive objects
978 SubIndex 15	List of drive objects
978 SubIndex 16	List of drive objects
978 SubIndex 17	List of drive objects
978 SubIndex 18	List of drive objects
978 SubIndex 19	List of drive objects
978 SubIndex 20	List of drive objects
978 SubIndex 21	List of drive objects
978 SubIndex 22	List of drive objects
978 SubIndex 23	List of drive objects
978 SubIndex 24	List of drive objects
979 SubIndex 0	PROFIdrive encoder format [0] = Header
979 SubIndex 1	PROFIdrive encoder format [1] = Type encoder 1
979 SubIndex 2	PROFIdrive encoder format [2] = Resolution enc 1
979 SubIndex 3	PROFIdrive encoder format [3] = Shift factor G1_XIST1
979 SubIndex 4	PROFIdrive encoder format [4] = Shift factor G1_XIST2
979 SubIndex 5	PROFIdrive encoder format [5] = Distinguishable revolutions encoder 1
979 SubIndex 6	PROFIdrive encoder format [6] = Reserved
979 SubIndex 7	PROFIdrive encoder format [7] = Reserved
979 SubIndex 8	PROFIdrive encoder format [8] = Reserved
979 SubIndex 9	PROFIdrive encoder format [9] = Reserved
979 SubIndex 10	PROFIdrive encoder format [10] = Reserved
979 SubIndex 11	PROFIdrive encoder format [11] = Type encoder 2
979 SubIndex 12	PROFIdrive encoder format [12] = Resolution enc 2
979 SubIndex 13	PROFIdrive encoder format [13] = Shift factor G2_XIST1
979 SubIndex 14	PROFIdrive encoder format [14] = Shift factor G2_XIST2
979 SubIndex 15	PROFIdrive encoder format [15] = Distinguishable revolutions encoder 2
979 SubIndex 16	PROFIdrive encoder format [16] = Reserved
979 SubIndex 17	PROFIdrive encoder format [17] = Reserved
979 SubIndex 18	PROFIdrive encoder format [18] = Reserved
979 SubIndex 19	PROFIdrive encoder format [19] = Reserved
979 SubIndex 20	PROFIdrive encoder format [20] = Reserved
979 SubIndex 21	PROFIdrive encoder format [21] = Type encoder 3
979 SubIndex 22	PROFIdrive encoder format [22] = Resolution enc 3
979 SubIndex 23	PROFIdrive encoder format [23] = Shift factor G3_XIST1
979 SubIndex 24	PROFIdrive encoder format [24] = Shift factor G3_XIST2
979 SubIndex 25	PROFIdrive encoder format [25] = Distinguishable revolutions encoder 3
979 SubIndex 26	PROFIdrive encoder format [26] = Reserved

Parameter	Parameter text
979 SubIndex 27	PROFIdrive encoder format [27] = Reserved
979 SubIndex 28	PROFIdrive encoder format [28] = Reserved
979 SubIndex 29	PROFIdrive encoder format [29] = Reserved
979 SubIndex 30	PROFIdrive encoder format [30] = Reserved
1082	Maximum speed / n_max
1520	Torque limit upper/motoring / M_max upper/mot
1521	Torque limit lower/regenerative / M_max lower/regen
1544	Travel to fixed stop evaluation torque reduction / TfS M_red eval
2000	Reference speed reference frequency / n_ref f_ref
2003	Reference torque / M_ref
2079	IF1 PROFIdrive PZD telegram selection extended / IF1 PZD telegr ext

This makes complete data adaptation possible.

DynamicServoControl – Simulation of a drive controller component in a PROFIdrive telegramm

Symbol



Function

The *DynamicServoControl* component type is used to enhance a PROFIdrive telegram according to the PROFIdrive profile (Technical Specification, Version 4.2, Edition: October 2015 (Order No.: 3.172)) by the functionality of the drive-internal controller.

	PZD 1	PZD 2	PZD 3	PZD 4	PZD 5	PZD 6	PZD 7	PZD 8	PZD 9
TEL_5 Nominal	STW 1	NSOLL_B		STW 2	G1_STW	XERR		KPC	
TEL_5 Actual	ZSW 1	NIST_B		ZSW 2	G1_ZSW	G1_XIST 1		G1_XIST 2	

Telegrams with a higher value have been defined in the PROFIdrive profile; these can be used to move the position control from the controller to the drive. This is particularly important when the technology application of the controller's cycle is reduced compared to the bus clock.

Moving the position control from the controller to the drive necessarily requires an isochronous bus system and that the SIMIT application is operated in isochronous mode. Set isochronous mode (Page 42) for this.

Connectors

Input	Description
XERR	Error signal from the PROFIdrive telegram
KPC	Controller gain from the PROFIdrive telegram
DriveIn	Structure for docking to upstream PROFIdrive components

Output	Description
DriveOut	Structure for connecting additional PROFIdrive components

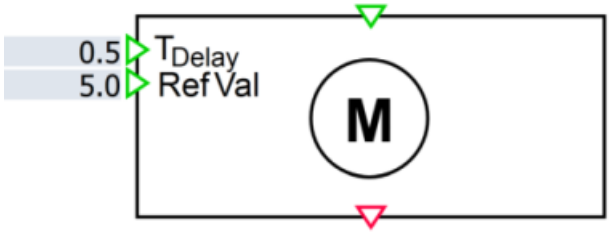
Parameter

Parameter	Description
G1_FineResolutionXist1	Enter the fine resolution of G1_Xist1 here.
G1_FineResolutionXist2	Enter the fine resolution of G1_Xist2 here.
G2_FineResolutionXist1	Enter the fine resolution of G2_Xist1 here.
G2_FineResolutionXist2	Enter the fine resolution of G2_Xist2 here.

The parameter values should be consistent with the parameters of the same name of the sensor block.

GeneralDrive

Symbol



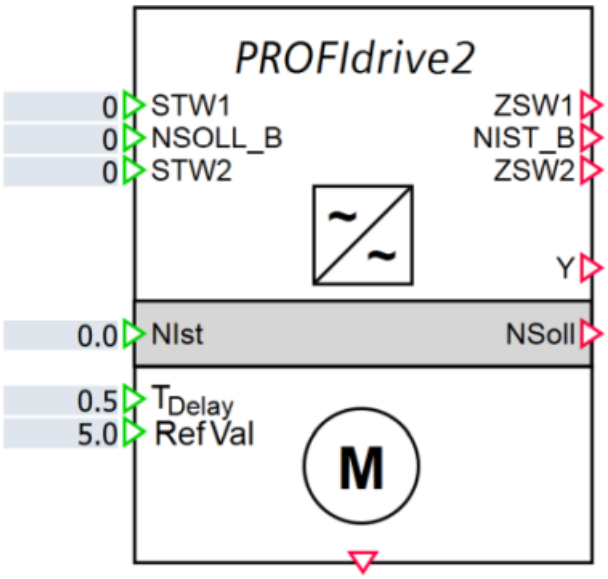
Function

The *GeneralDrive* component type supplements the PROFIdrive block with the functionality of a delay element. The *actual speed* is calculated internally on the basis of the specified setpoint, so that an external connection of the NIst input on the PROFIdrive component is no longer necessary.

T_{Delay} is the delay with which the speed actual value tracks the effective setpoint value (ramp function generator value from PROFIdrive block) via the delay function. The default delay time is half a second.

The reference value at analog input *RefVal* is a percentage speed value. If the speed actual value exceeds the reference value, bit 10 of the status word (ZSW1.10) is set to 1. The default reference value is 5 percent. The switchover of this signal is performed using a configurable

hysteresis. This is also stated as a percentage speed value in the *Hysteresis* parameter. The default hysteresis value is 3 percent.



Connectors

Input	Description
T_{Delay}	Time constant of the delay time
RefVal	Reference value set to one with bit 10 in status word (ZSW1.10)
DriveIn	Structure for docking to upstream PROFIdrive components

Output	Description
DriveOut	Structure for connecting additional PROFIdrive components

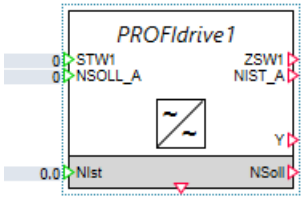
Parameter

Parameters	Description
ReferenceSpeed	Here you enter the reference speed.
RampTime	Here you enter the ramp slope for changing the speed setpoint.
EmergencyStopTime	Here you enter the slope with which the pending speed setpoint is to be reduced in case of an error.

HysteresisSpeed	Enter here the width of the window which will be moved around the target speed. If the actual speed is within this window, ZSW1.Bit8 (Speed Error Within Tolerance) is set to TRUE.
SpeedThreshold	Enter a comparison value for the speed threshold here. If the actual speed has reached or exceeded this comparison value, ZSW1.Bit10 (Speed Reached) is set to TRUE.

PROFIdrive1 – Basic function of the PROFIdrive drive for speed-controlled axes

Symbol



Function

The *PROFIdrive1* component type simulates the state machine with the ramp function generator according to the PROFIdrive profile (Technical Specification, Version 4.2, Edition: October 2015 (Order No.: 3.172)).

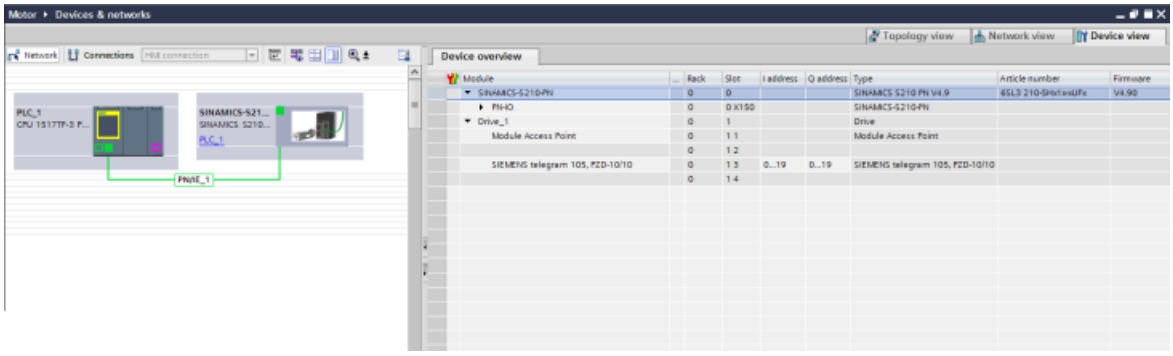
Functionalities for Application Class 1 (AC1) "Standard Drive – Speed Control Mode" are simulated.

This component therefore simulates the functionality defined in standard PROFIdrive telegram 1:

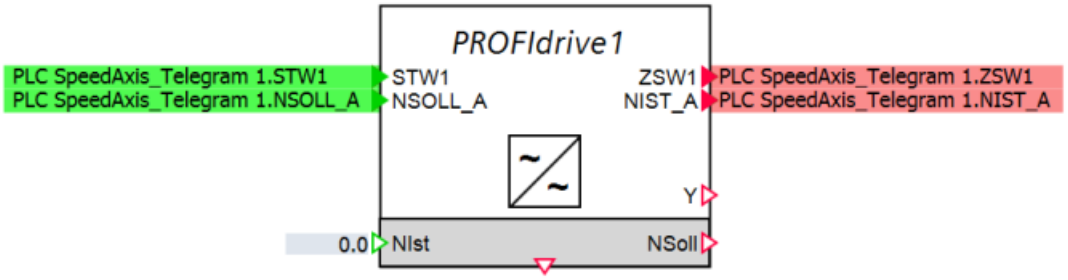
	PZD 1	PZD 2
TEL_1 Nominal	STW 1	NSOLL_A
TEL_1 Actual	ZSW 1	NIST_A

The starting point for configuring a *PROFIdrive1* component is the controller access to the input and output signals of the PROFIdrive drive. For each dive, a fixed address range is defined within the controller input and output area, and assigned to a PROFIdrive1 component as follows: The control word (first process data word in the output area) is connected to the integer input *STW1* and the status word (first process data word of the controller inputs) is connected to the integer output *ZSW1*. The speed setpoint and actual speed values are transferred in the second process data word. Accordingly, the integer input *NSOLL_A* is connected to the speed setpoint value. The integer output *NIST_A* is connected to the speed actual value of the controller.

In the figure below, an example of a SIMATIC configuration for SINAMICS is shown.



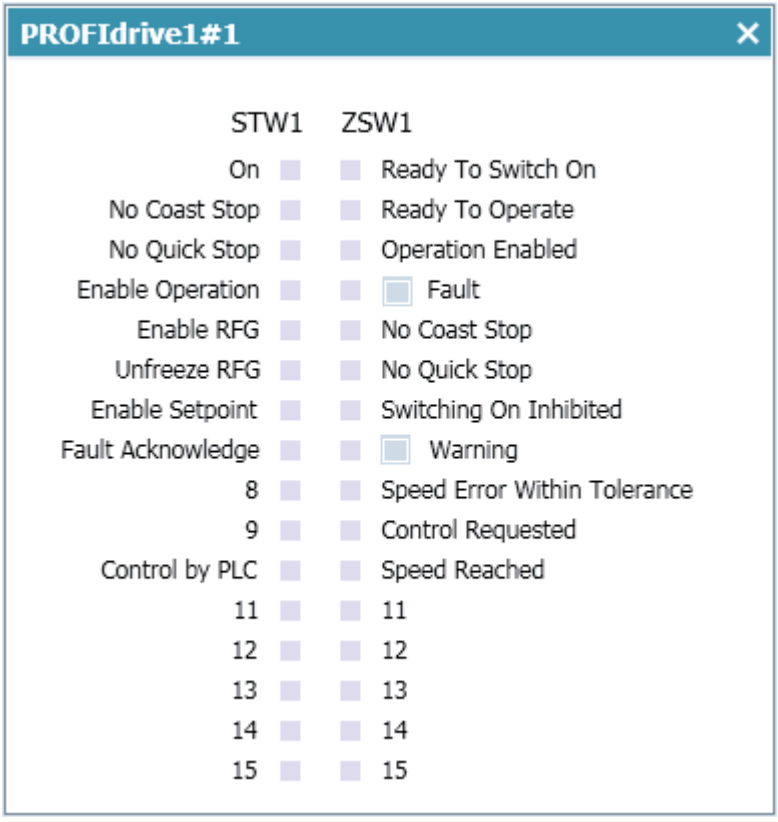
The figure below shows the *PROFIdrive1* component and the connected process data:



The process data speed values, which means *NSOLL_A* and *NIST_A* are raw values in the range zero to 16384. The value 16384 corresponds to the nominal speed. Other process data, for example, torque values or actual current values, are not taken into account in the *PROFIdrive1* component.

The current speed of the drive is output as a percentage at analog output *Y*, in other words in a range of 0 to 100: $0 \leq Y \leq 100$. A value of 0 thus corresponds to a stationary drive, while a value 100 corresponds to the nominal speed (raw speed value 16384).

The contents of the control word and the status word are displayed bit-by-bit in the operating window of the component. In addition, the setpoint speed value *NSOLL_A* and the speed actual value *NIST_A*, are displayed as percentage values and in revolutions per minute (*RPM*). The fourth bit and eighth bit in the status word (ZSW1.3 - *Fault* and ZSW1.7 - *Warning*) can be set to 1 using a toggle switch.

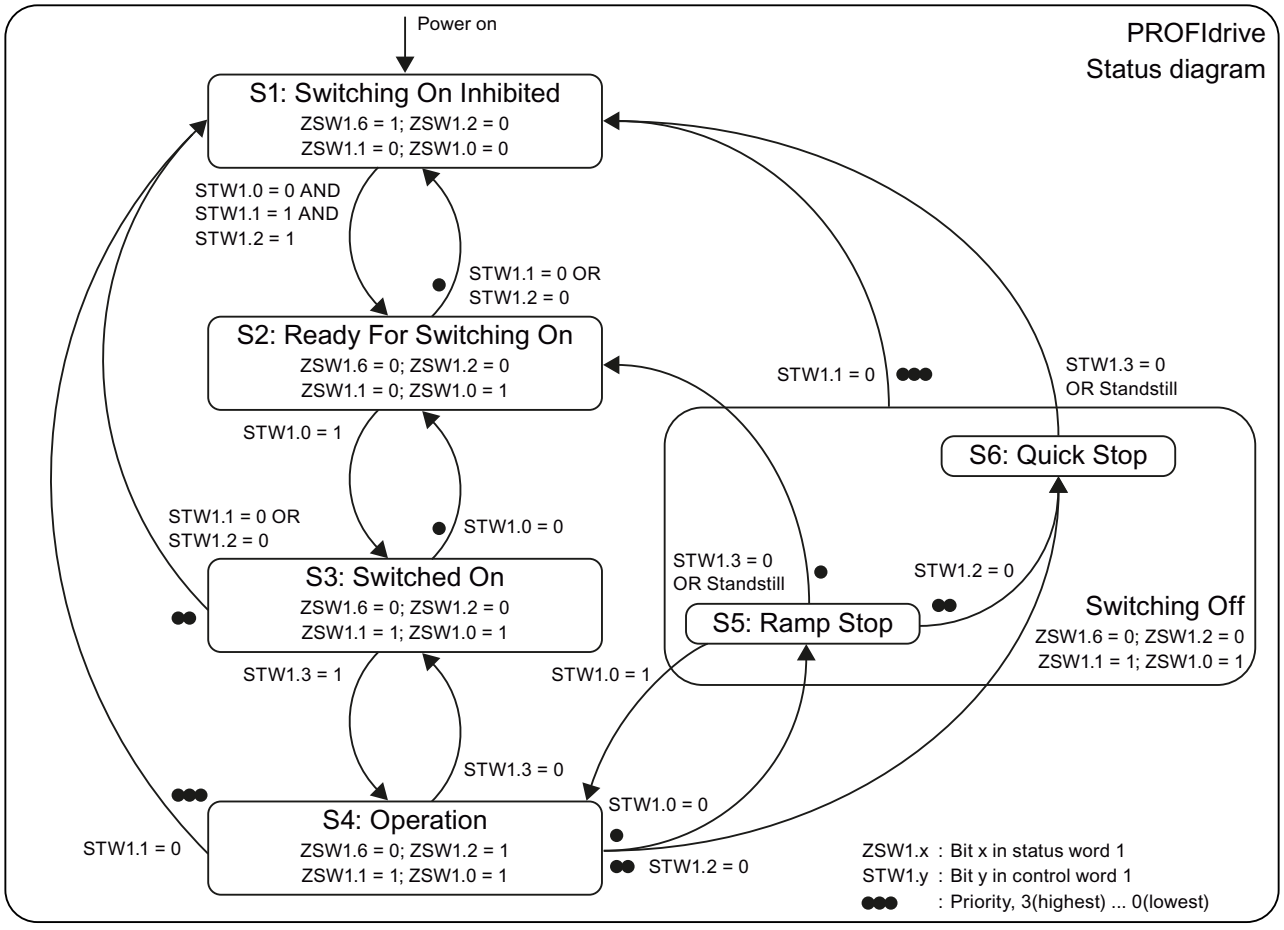


See also

Using the PROFIdrive library to simulate PROFIdrive drives (Page 512)

State machine

The state machine is described in detail in the PROFIdrive specification. The state graph implemented in the *PROFIdrive1* component type is illustrated in the figure below:



The control bits shown in the table are used to control the state machine (status transitions). Depending on the current status, the corresponding status bits are set as illustrated in the table.

Table 8-8 Status table for the component PROFIdrive1

State	State	Description	ZSW1 (bit sequence 15 ... 0)
S1: Switching On Inhibited	1	Off	dddd dmsm a1ss a000
S2: Ready For Switching On	2	Ready	dddd dmsm a0ss a001
S3: Switched On	3	Switched on	dddd dmsm a0ss a011
S4: Operation	4	Normal operation	dddd dmsm a0ss a111
S5: Switching Off (Ramp Stop)	5	Ramp stop of the motor	dddd dmsm a0ss a011
S6: Switching Off (Quick Stop)	6	Quick stop of the motor	dddd dmsm a0ss a011

The individual bits in status word *ZSW1* shown in the table have the following meaning:

d device-specific (device-specific component)
s derived directly from *STW1*
m model-specific (here: delay)
a can be set in the operating window

Table 8-9 Structure of control word

Name	STW1.Bit	Description	Use
On/Off (Off1)	STW1.0	Switch off drive	Yes
No Coast Stop (Off2)	STW1.1	No coast stop of drive	Yes
No Quick Stop (Off3)	STW1.2	No quick stop of drive	Yes
Enable Operation	STW1.3	Drive moves to setpoint	Yes
Enable Ramp Generator	STW1.4	RFG is used	Yes
Unfreeze Ramp Generator	STW1.5	RFG frozen	Yes
Enable Setpoint	STW1.6	NSOLL as input for RFG	Yes
Fault Acknowledge	STW1.7	Fault acknowledgement from PLC	Yes
Jog 1 On	STW1.8	not implemented	No
Jog 2 On	STW1.9	not implemented	No
Control by PLC	STW1.10	DO IO data valid	Yes
Device-specific	STW1.11–15		Device-specific component

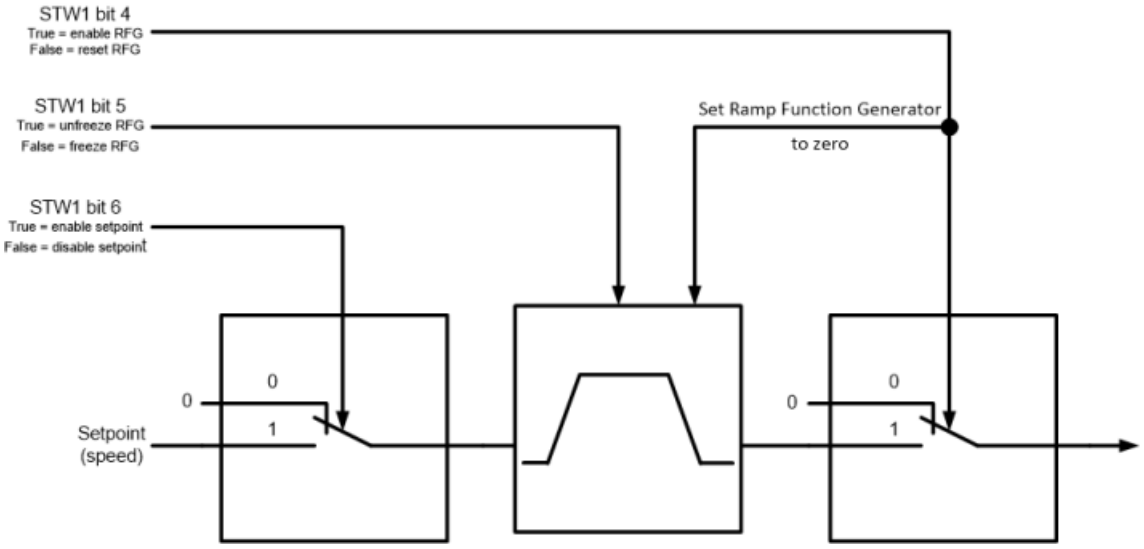
The meaning of all the status word bits is described in the table below.

Table 8-10 Structure of the status word

Name	ZSW1.Bit	Description	Source
Ready To Switch On	ZSW1.0	Power supply switched on	State machine
Ready To Operate	ZSW1.1	No fault present	State machine
Operation Enabled	ZSW1.2	Operation enabled	State machine
Fault Present	ZSW1.3	Drive faulty/out of service	Operating window
Coast Stop Not Active	ZSW1.4	No coast stop of drive	STW1.1
Quick Stop Not Active	ZSW1.5	No quick stop of drive	STW1.2
Switching On Inhibited	ZSW1.6	Switching on inhibited	State machine
Warning Present	ZSW1.7	Warning present/out of service	Operating window
Speed Error Within Tolerance	ZSW1.8	Setpoint/actual deviation in tolerance range	PT1
Control Requested	ZSW1.9	Control requested/local operation	STW1.10
Speed Reached	ZSW1.10	Reference speed reached	PT1
Device-specific	ZSW1.11–15	Drive-specific	Additional component/0

Ramp generator

The ramp function generator (RFG) is controlled by bits 4, 5 and 6 of the control word (STW1.4, STW1.5, STW1.6). Bit 6 sets the input of the generator to *NSOLL_RFG*. The generator then ramps up or down to this setpoint value in a linear manner using the set time *RampTime* assuming the generator is enabled via bit 5. Otherwise, the last output value of the generator is retained. Finally, bit 4 determines whether the ramp generator value is output. A setpoint connection can only be carried out in state S4. Otherwise the ramp setpoint 0 is output.



Specific output for expansion

A specific Drive connector on the bottom of the component is used to connect a device-specific component or for adding additional functions.

This connector is of the PROFIdrive connection type. It cannot be connected to the analog, binary or integer connectors of components.

Use of the PROFIdrive1 component type without extension

The *PROFIdrive1* component type implements the standard functions according to the PROFIdrive profile. The *PROFIdrive1* component can therefore be used to simulate PROFIdrive devices with this basic function, without being connected to a device-specific component. Drive-specific bits in the control word are then ignored and drive-specific bits in the status word are set to zero.

Connectors

Input	Description
STW1	Control word 1 (STW1) from the PROFIdrive telegram
NSOLL_A	Standardized speed setpoint (NSOLL_A) from the PROFIdrive telegram
NIst	Process value actual speed

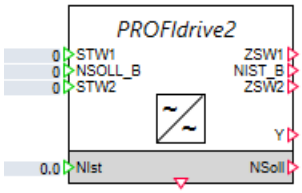
Output	Description
ZSW1	Status word 1 (ZSW1) from the PROFIdrive telegram
NIST_A	Standardized actual speed value (NIST_A) from the PROFIdrive telegram
Y	Process value speed setpoint value specification relative to the reference speed in %
NSoll	Process value speed setpoint value specification
DriveOut	Structure for connecting additional PROFIdrive components

Parameter

Parameters	Description
ReferenceSpeed	Here you enter the reference speed.
MaxSpeed	Here you enter the maximum physical speed of the drive.
RampTime	Here you enter the ramp slope for changing the speed set-point.
EmergencyStopTime	Here you enter the slope with which the pending speed set-point is to be reduced in case of an error.
HysteresisSpeed	Enter here the width of the window which will be moved around the target speed. If the actual speed is within this window, ZSW.Bit8 (Speed Error Within Tolerance) is set to TRUE.
SpeedThreshold	Enter a comparison value for the speed threshold here. If the actual speed has reached or exceeded this comparison value, ZSW.Bit10 (Speed Reached) is set to TRUE.

PROFIdrive2 – Basic function of the PROFIdrive drive for positioning axes

Symbol



Function

The PROFIdrive2 component type simulates the state machine with the ramp function generator according to the PROFIdrive profile (Technical Specification, Version 4.2, Edition: October 2015 (Order No.: 3.172)).

In addition to the PROFIdrive1 block, the PROFIdrive2 block provides a monitoring option in form of a sign-of-life signal exchange in STW2 and ZSW2.

The cyclic exchange of the sign-of-life requires that the SIMIT application is operated in isochronous mode. Set isochronous mode for this.

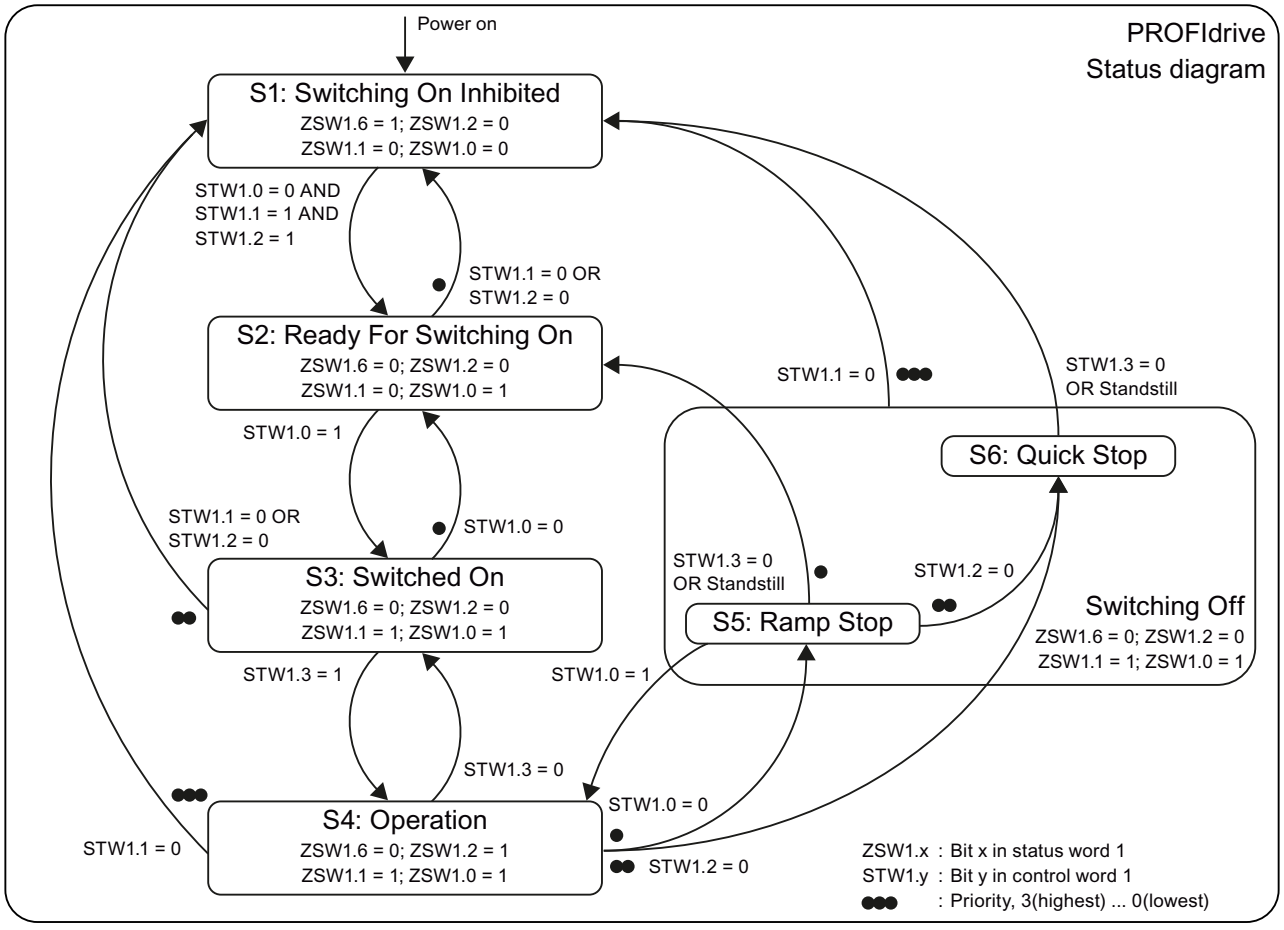
This component therefore simulates the functionality defined in the standard PROFIdrive telegram 2 that is used for operation of extended speed-controlled axes:

	PZD 1	PZD 2	PZD 3	PZD 4
TEL_2 Nominal	STW 1	NSOLL_B		STW 2
TEL_2 Actual	ZSW 1	NIST_B		ZSW 2

Additional components are required for modeling positioning axes; these can be docked over the special connector for expansion (see also Specific output for expansion (Page 533)).

State machine

The state machine is described in detail in the PROFIdrive specification. The state graph implemented in the *PROFIdrive2* component type is illustrated in the figure below:



The control bits shown in the table are used to control the state machine (status transitions). Depending on the current status, the corresponding status bits are set as illustrated in the table.

Table 8-11 Status table for the component PROFIdrive2

State	State	Description	ZSW1 (bit sequence 15 ... 0)
S1: Switching On Inhibited	1	Off	dddd dmsm a1ss a000
S2: Ready For Switching On	2	Ready	dddd dmsm a0ss a001
S3: Switched On	3	Switched on	dddd dmsm a0ss a011
S4: Operation	4	Normal operation	dddd dmsm a0ss a111
S5: Switching Off (Ramp Stop)	5	Ramp stop of the motor	dddd dmsm a0ss a011
S6: Switching Off (Quick Stop)	6	Quick stop of the motor	dddd dmsm a0ss a011

The individual bits in status word *ZSW1* shown in the table have the following meaning:

d device-specific (device-specific component)
s derived directly from *STW1*
m model-specific (here: delay)
a can be set in the operating window

Table 8-12 Structure of control word

Name	STW1.Bit	Description	Use
On/Off (Off1)	STW1.0	Switch off drive	Yes
No Coast Stop (Off2)	STW1.1	No coast stop of drive	Yes
No Quick Stop (Off3)	STW1.2	No quick stop of drive	Yes
Enable Operation	STW1.3	Drive moves to setpoint	Yes
Enable Ramp Generator	STW1.4	RFG is used	Yes
Unfreeze Ramp Generator	STW1.5	RFG frozen	Yes
Enable Setpoint	STW1.6	NSOLL as input for RFG	Yes
Fault Acknowledge	STW1.7	Fault acknowledgement from PLC	Yes
Jog 1 On	STW1.8	not implemented	No
Jog 2 On	STW1.9	not implemented	No
Control by PLC	STW1.10	DO IO data valid	Yes
Device-specific	STW1.11–15		Device-specific component

The meaning of all the status word bits is described in the table below.

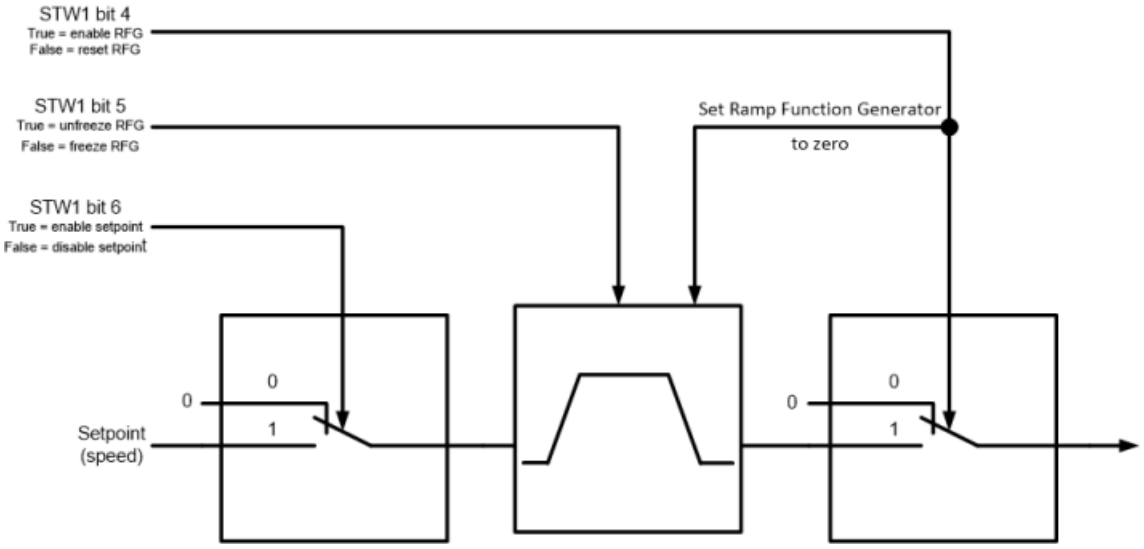
Table 8-13 Structure of the status word

Name	ZSW1.Bit	Description	Source
Ready To Switch On	ZSW1.0	Power supply switched on	State machine
Ready To Operate	ZSW1.1	No fault present	State machine
Operation Enabled	ZSW1.2	Operation enabled	State machine

Name	ZSW1.Bit	Description	Source
Fault Present	ZSW1.3	Drive faulty/out of service	Operating window
Coast Stop Not Active	ZSW1.4	No coast stop of drive	STW1.1
Quick Stop Not Active	ZSW1.5	No quick stop of drive	STW1.2
Switching On Inhibited	ZSW1.6	Switching on inhibited	State machine
Warning Present	ZSW1.7	Warning present/out of service	Operating window
Speed Error Within Tolerance	ZSW1.8	Setpoint/actual deviation in tolerance range	PT1
Control Requested	ZSW1.9	Control requested/local operation	STW1.10
Speed Reached	ZSW1.10	Reference speed reached	PT1
Device-specific	ZSW1.11–15	Drive-specific	Additional component/0

Ramp generator

The ramp function generator (RFG) is controlled by bits 4, 5 and 6 of the control word (STW1.4, STW1.5, STW1.6). Bit 6 sets the input of the generator to *NSOLL_RFG*. The generator then ramps up or down to this setpoint value in a linear manner using the set time *RampTime* assuming the generator is enabled via bit 5. Otherwise, the last output value of the generator is retained. Finally, bit 4 determines whether the ramp generator value is output. A setpoint connection can only be carried out in state S4. Otherwise the ramp setpoint 0 is output.



Sign-of-life exchange over STW2 and ZSW2

The process data STW2 and ZSW2 are defined in the standard PROFIdrive telegrams 2 and higher. They are used to exchange a sign-of-life between controller and drive. If the sign-of-life fails in a configurable sequence, an error is generated and the drive is shut down for safety reasons.

Specific output for expansion

A specific Drive connector on the bottom of the component is used to connect a device-specific component or for adding additional functions.

This connector is of the PROFIdrive connection type. It cannot be connected to the analog, binary or integer connectors of components.

Use of the PROFIdrive2 component type without extension

The *PROFIdrive2* component type implements the standard functions according to the PROFIdrive profile. The *PROFIdrive2* component can therefore be used to simulate PROFIdrive devices with this basic function, without being connected to a device-specific component. Drive-specific bits in the control word are then ignored and drive-specific bits in the status word are set to zero.

If higher-value PROFIdrive telegrams (3, 4, 5, 6, 102, 103, 105, 106) are to be modeled, the basic block must be expanded accordingly.

Connectors

Input	Description
STW1	Control word 1 (STW1) from the PROFIdrive telegram
NSOLL_B	Standardized speed setpoint (NSOLL_B) from the PROFIdrive telegram
STW2	Control word 2 (STW2) from the PROFIdrive telegram
NIst	Process value actual speed

Output	Description
ZSW1	Status word 1 (ZSW1) from the PROFIdrive telegram
NIST_B	Standardized speed actual value (NIST_B) from the PROFIdrive telegram
ZSW2	Status word 2 (ZSW2) from the PROFIdrive telegram
Y	Process value speed setpoint value specification relative to the reference speed in %
NSoll	Process value speed setpoint value specification
DriveOut	Structure for connecting additional PROFIdrive components

Parameter

Parameters	Description
ReferenceSpeed	Here you enter the reference speed.
MaxSpeed	Here you enter the maximum physical speed of the drive.
RampTime	Here you enter the ramp slope for changing the speed set-point.
EmergencyStopTime	Here you enter the slope with which the pending speed set-point is to be reduced in case of an error.
AllowedLifeSignErrors	Here you enter the number of tolerated sign-of-life failures. (0 means that not a single sign-of-life may be missed)

HysteresisSpeed	Enter here the width of the window which will be moved around the target speed. If the actual speed is within this window, ZSW1.Bit8 (Speed Error Within Tolerance) is set to TRUE.
SpeedThreshold	Enter a comparison value for the speed threshold here. If the actual speed has reached or exceeded this comparison value, ZSW1.Bit10 (Speed Reached) is set to TRUE.

Safety30 – Enhancement of the drive simulation with a PROFISafe telegram 30

Symbol



Function

The *Safety30* component type is used to enhance a PROFIdrive telegram according to the PROFIdrive profile (Technical Specification, Version 4.2, Edition: October 2015 (Order No.: 3.172)) by the functionality of Safety telegram 30.

This block processes user data of Safety telegram 30. The simulation of the PROFISafe trailer is not performed. Simulation of the secure PROFISafe communication is not required to use PLCSIM_Adv.

Connectors

Input	Description
S_STW1	Safety control word 1
DriveIn	Structure for docking to upstream PROFIdrive components

Output	Description
S_ZSW1	Safety status word 1
DriveOut	Structure for connecting additional PROFIdrive components

Parameter

Parameter	Description
SafetyTimeout	Monitoring time of an active Safety function in ms
STO_Enabled	Support for the Safety Integrated Function Safe Torque Off (STO)
SS1_Enabled	Support for the Safety Integrated Function Safe Stop 1 (SS1)
SS2_Enabled	Support for the Safety Integrated Function Safe Stop 2 (SS2)
SOS_Enabled	Support for the Safety Integrated Function Safe Operating Stop (SOS)
SLS_Enabled	Support for the Safety Integrated Function Safely-Limited Speed (SLS)
SLT_Enabled	Support for the Safety Integrated Function Safely-Limited Torque (SLT)

Parameter	Description
SLP_Enabled	Support for the Safety Integrated Function Safely-Limited Position (SLP)
SLA_Enabled	Support for the Safety Integrated Function Safely-Limited Acceleration (SLA)
SDI_Enabled	Support for the Safety Integrated Function Safely – Safe Direction (SDI)

Safety31 – Enhancement of the drive simulation with a PROFISafe telegram 31

Symbol



Function

The *Safety31* component type is used to enhance a PROFIdrive telegram according to the PROFIdrive profile (Technical Specification, Version 4.2, Edition: October 2015 (Order No.: 3.172)) by the functionality of Safety telegram 31.

This block processes user data of Safety telegram 31. The simulation of the PROFISafe trailer is not performed. Simulation of the secure PROFISafe communication is not required to use PLCSIM_Adv.

Connectors

Input	Description
S_STW2	Safety control word 2
DriveIn	Structure for docking to upstream PROFIdrive components

Output	Description
S_ZSW2	Safety status word 2
DriveOut	Structure for connecting additional PROFIdrive components

Parameter

Parameter	Description
SafetyTimeout	Monitoring time of an active Safety function in ms
STO_Enabled	Support for the Safety Integrated Function Safe Torque Off (STO)
SS1_Enabled	Support for the Safety Integrated Function Safe Stop 1 (SS1)
SS2_Enabled	Support for the Safety Integrated Function Safe Stop 2 (SS2)
SOS_Enabled	Support for the Safety Integrated Function Safe Operating Stop (SOS)
SLS_Enabled	Support for the Safety Integrated Function Safely-Limited Speed (SLS)
SLT_Enabled	Support for the Safety Integrated Function Safely-Limited Torque (SLT)

Parameter	Description
SLP_Enabled	Support for the Safety Integrated Function Safely-Limited Position (SLP)
SLA_Enabled	Support for the Safety Integrated Function Safely-Limited Acceleration (SLA)
SDI_Enabled	Support for the Safety Integrated Function Safely – Safe Direction (SDI)

Safety32 – Enhancement of the drive simulation with a PROFISafe telegram 32

Symbol



Function

The *Safety32* component type is used to enhance a PROFIdrive telegram according to the PROFIdrive profile (Technical Specification, Version 4.2, Edition: October 2015 (Order No.: 3.172)) by the functionality of Safety telegram 32.

This block processes user data of Safety telegram 32. The simulation of the PROFISafe trailer is not performed. Simulation of the secure PROFISafe communication is not required to use PLCSIM_Adv.

Connectors

Input	Description
S_STW2	Safety control word 2
DriveIn	Structure for docking to upstream PROFIdrive components

Output	Description
S_ZSW2	Safety status word 2
S_XIST32	Safety position of current value (for Safety telegram 32)
DriveOut	Structure for connecting additional PROFIdrive components

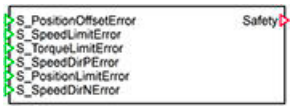
Parameter

Parameter	Description
SafetyTimeout	Monitoring time of an active Safety function in ms
STO_Enabled	Support for the Safety Integrated Function Safe Torque Off (STO)
SS1_Enabled	Support for the Safety Integrated Function Safe Stop 1 (SS1)
SS2_Enabled	Support for the Safety Integrated Function Safe Stop 2 (SS2)
SOS_Enabled	Support for the Safety Integrated Function Safe Operating Stop (SOS)
SLS_Enabled	Support for the Safety Integrated Function Safely-Limited Speed (SLS)

Parameter	Description
SLT_Enabled	Support for the Safety Integrated Function Safely-Limited Torque (SLT)
SLP_Enabled	Support for the Safety Integrated Function Safely-Limited Position (SLP)
SLA_Enabled	Support for the Safety Integrated Function Safely-Limited Acceleration (SLA)
SDI_Enabled	Support for the Safety Integrated Function Safely – Safe Direction (SDI)
G1_EncoderSystem	Enter the type of encoder system here. 0 = rotary encoder system 1 = linear encoder system

SafetyProcess – Resolution of errors during active Safety functions

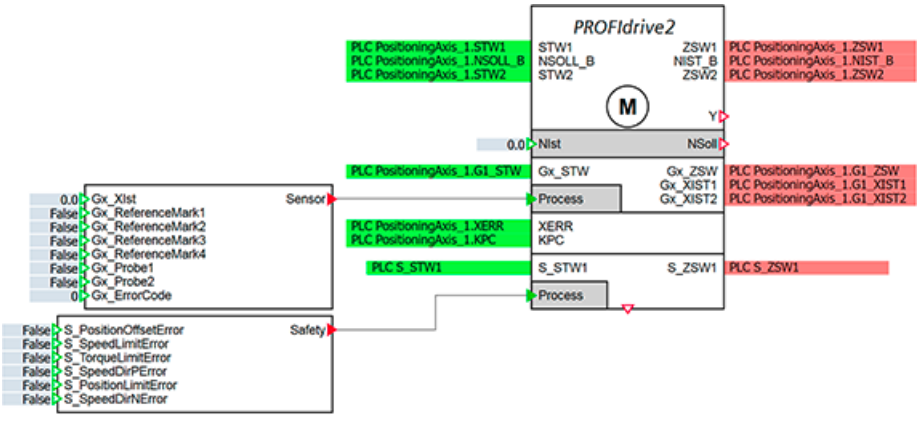
Symbol



Function

The *SafetyProcess* component type is used to resolve errors during active drive-internal Safety functions.

The component must be connected to the Process connector of a Safety component from the PROFIdrive telegram.



Connectors

Input	Description
S_PositionOffsetError	Resolution of a position monitoring error during an active Safety Integrated function: Safe Operating Stop (SOS)
S_SpeedLimitError	Resolution of a speed monitoring error during an active Safety Integrated function: Safely-Limited Speed (SLS)

Input	Description
S_TorqueLimitError	Resolution of a torque limit violation during an active Safety Integrated function: Safely-Limited Torque (SLT)
S_SpeedDirPError	Resolution of a direction reversal (negative) during an active Safety Integrated function: Safe Direction (SDI)
S_PositionLimitError	Resolution of a position limit violation during an active Safety Integrated function: Safely Limited Position (SLP)
S_SpeedDirNError	Resolution of a direction reversal (positive) during an active Safety Integrated function: Safe Direction (SDI)

Output	Description
Safety	Output structure for the interconnection to the Process connector of a Safety component from the PROFIdrive telegram.

Sensor – Simulation of the sensor component in a PROFIdrive telegram

Symbol



Function

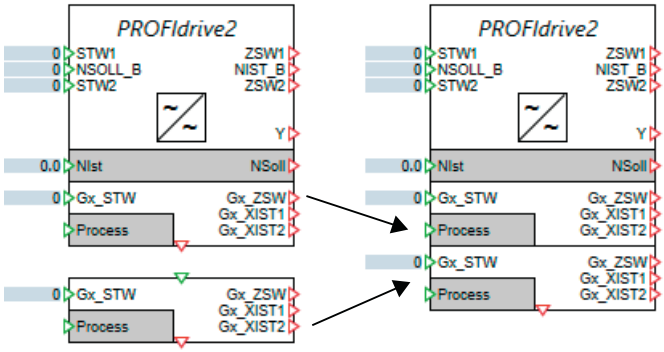
The *Sensor* component type is used to enhance a PROFIdrive telegram according to the PROFIdrive profile (Technical Specification, Version 4.2, Edition: October 2015 (Order No.: 3.172)) by the functionality of a sensor.

This component therefore simulates the part of a sensor in the standard PROFIdrive telegram:

	PZD 1	PZD 2	PZD 3	PZD 4	PZD 5	PZD 6	PZD 7	PZD 8	PZD 9
TEL_3 Nominal	STW 1	NSOLL_B		STW 2	G1_STW				
TEL_3 Actual	ZSW 1	NIST_B		ZSW 2	G1_ZSW	G1_XIST 1		G1_XIST 2	

If a telegram is to be simulated with two sensors, this can be done by docking an additional "Sensor" component.

	PZD 1	PZD 2	PZD 3	PZD 4	PZD 5	PZD 6	PZD 7	PZD 8	PZD 9	PZD 10	PZD 11	PZD 12	PZD 13	PZD 14
TEL_4 Nominal	STW 1	NSOLL_B		STW 2	G1_STW	G2_STW								
TEL_4 Actual	ZSW 1	NIST_B		ZSW 2	G1_ZSW	G1_XIST 1		G1_XIST 2		G2_ZSW	G2_XIST 1		G2_XIST 2	



Connectors

Input	Description
Gx_STW	Sensor control word (Gx_STW) from the PROFIdrive telegram
Process	Structure for docking a sensor simulation
DriveIn	Structure for docking to upstream PROFIdrive components

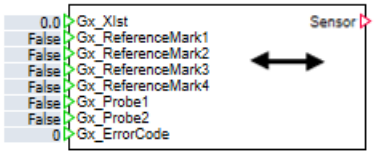
Output	Description
Gx_ZSW	Sensor status word (Gx_ZSW) from the PROFIdrive telegram
Gx_XIST1	Sensor actual value 1 (Gx_XIST1) from the PROFIdrive telegram
Gx_XIST2	Sensor actual value 2 (Gx_XIST2) from the PROFIdrive telegram
DriveOut	Structure for connecting additional PROFIdrive components

Parameter

Parameters	Description
GxFineResolutionXist1	Enter the fine resolution of Gx_Xist1 here.
GxFineResolutionXist2	Enter the fine resolution of Gx_Xist2 here.

SensorProcessLinear – Simulation of a linear encoder

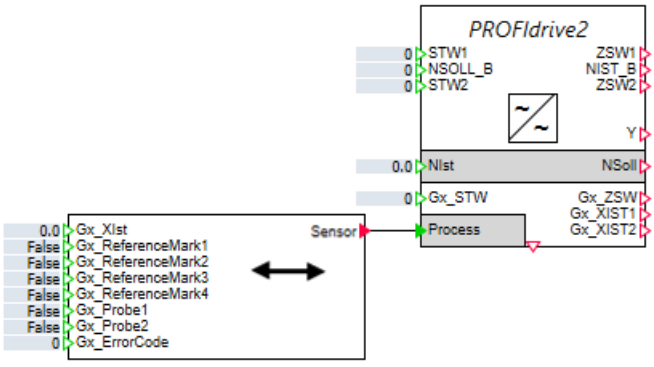
Symbol



Function

The *SensorProcessLinear* component type is used to simulate the functionality of a linear encoder. You can simulate linear absolute encoders as well as linear incremental encoders.

The component must be connected to the ""Process"" connector of a "Sensor" (Page 538) component from the PROFIdrive telegram.



Connectors

Input	Description
Gx_XIST	Process value in the engineering unit mm
Gx_ReferenceMark1...4	Encoder reference marks 1–4 (TRUE = reference mark deflected)
Gx_Probe1..2	Measuring probe deflection 1–2 (TRUE = measuring probe deflected)
Gx_ErrorCode	Simulated encoder error code

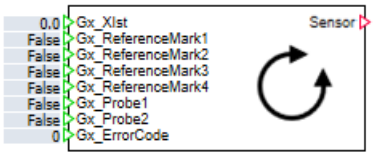
Output	Description
Sensor	Output structure for connection to the "Process" connector of a "Sensor" (Page 538) component from the PROFIdrive telegram.

Parameter

Parameter	Description
DistanceBetweenIncrements	Here you enter the distance between two tick marks in millimeters.
SupportsAbsoluteValues	Here you enter whether an absolute (=TRUE) or an incremental (=FALSE) encoder is to be simulated.

SensorProcessRotatory – Simulation of a rotary encoder

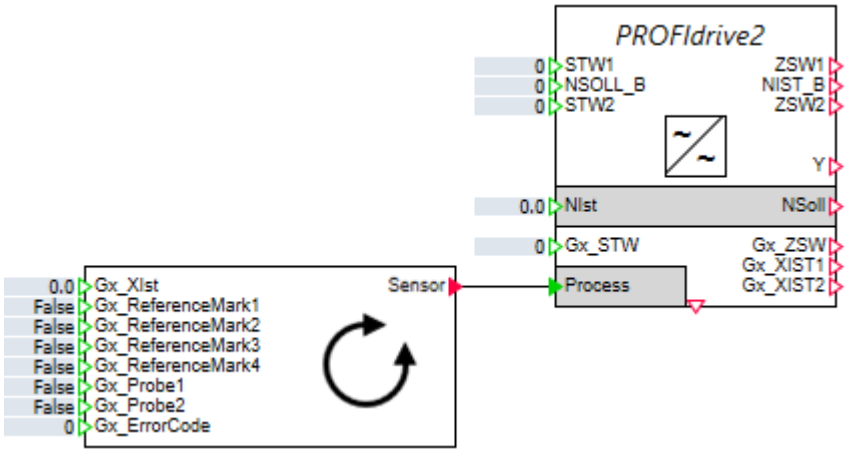
Symbol



Function

The *SensorProcessRotatory* component type is used to simulate the functionality of a rotary encoder. You can simulate rotary absolute encoders as well as rotary incremental encoders.

The component must be connected to the "Process" connector of a "Sensor" (Page 538) component from the PROFIdrive telegram.



Connectors

Input	Description
Gx_XIST	Process value of a physical variable
Gx_ReferenceMark1...4	Encoder reference marks 1–4 (TRUE = reference mark deflected)
Gx_Probe1..2	Measuring probe deflection 1–2 (TRUE = measuring probe deflected)
Gx_ErrorCode	Simulated encoder error code

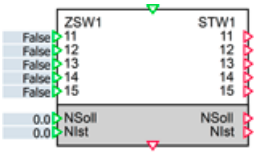
Output	Description
Sensor	Output structure for connection to the ""Process"" connector of a "Sensor" (Page 538) component from the PROFIdrive telegram.

Parameters

Parameter	Description
IncrementsPreRevolution	Here you enter the encoder property encoder line per revolution.
DeterminableRevolutions	Here you enter the number of revolutions for rotary encoders in which the absolute value can be determined.
DistancePerRevolution	Here you enter the distance in the physical unit you have selected that is covered in one encoder revolution (e.g. 360.0 for 360°).
SupportsAbsoluteValues	Here you enter whether an absolute (=TRUE) or an incremental (=FALSE) encoder is to be simulated.

Universal – Extensions to the PROFIdrive basic function

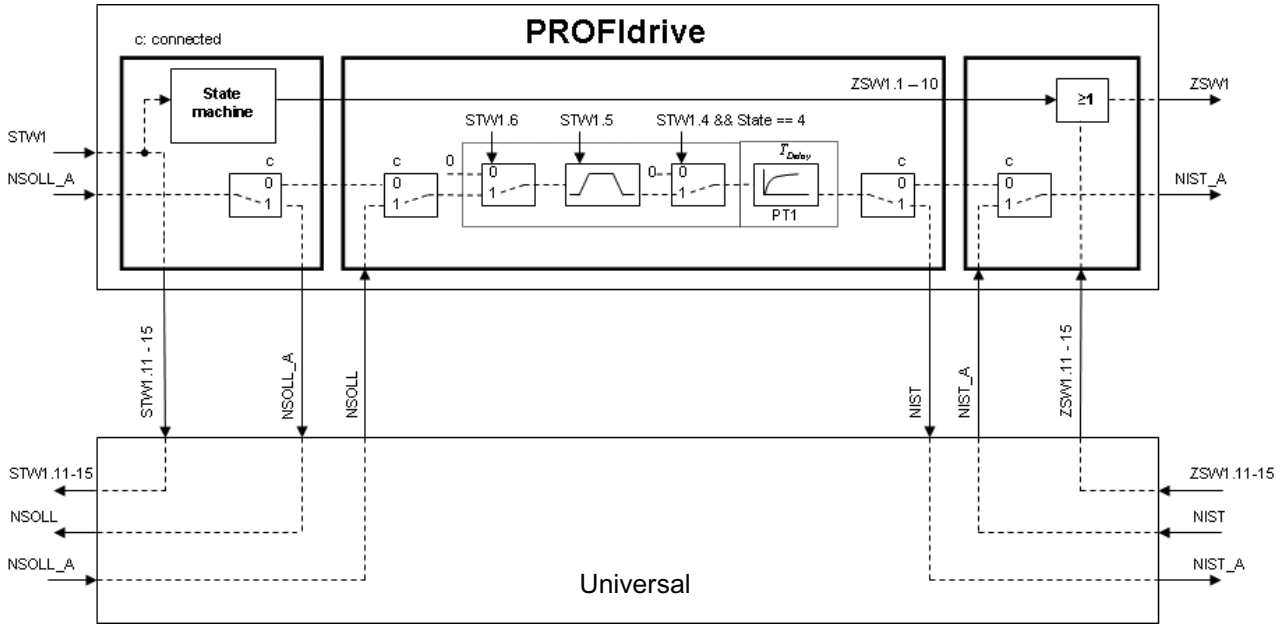
Symbol



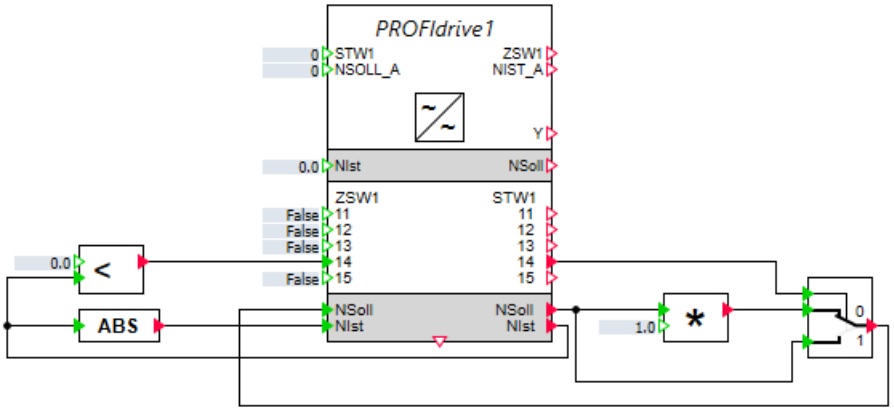
Function

The *Universal* component type can only be meaningfully used in combination with the *PROFIdrive1* or *PROFIdrive2* component type. It enables drive-specific functions to be realized in addition to the basic PROFIdrive functions.

Bits 11 to 15 of the control word (STW1.11–15), the speed setpoint value *NSOLL_A* and the speed actual value *NIST (NIST_PT1)* are placed on the outputs of the *Universal* component. By means of appropriate logical and arithmetical operations, bits 11 to 15 of the status word, the speed setpoint value *NSOLL* (*NSOLL_RFG*), and the speed actual value *NIST_A* are set at the inputs of the component. The signal linkage resulting from the connection of the *Universal*component to the *PROFIdrive* component is illustrated in the figure below.



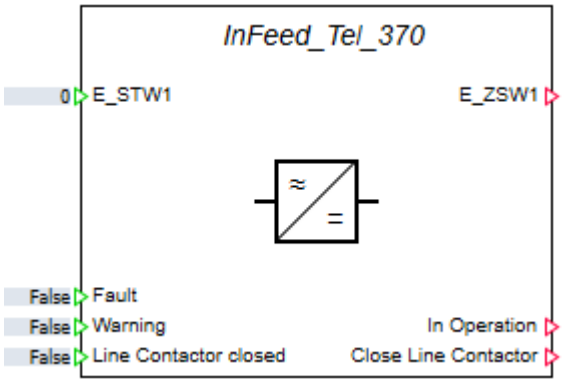
The figure below illustrates an example of how the drive-specific functions for the Type 3 Micromaster can be implemented with the aid of the *Universal* component.



Siemens

InFeed_Tel_370

Symbol



Function

The *Infeed_Tel_370* component type simulates an electrical infeed unit that is switched on and off by commands that are transported by the PROFIdrive telegram 370.

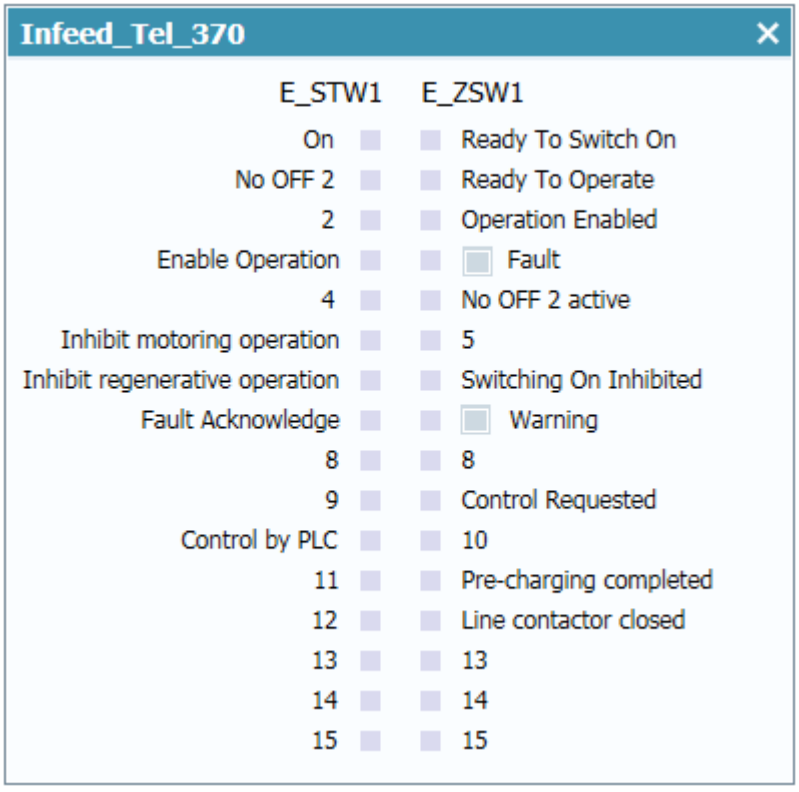
Telegram structure p922=370

	Output data		**Input data**	
Telegram 370	Signal	Comment	Signal	Comment
PZD1	E_STW1		E_ZSW1	

To signal a warning or error situation, set the binary input *Warning* or *Fault* from False to True.

When the binary input *Close Line Contactor* is set to True, the electrical infeed unit requests that the line contactor is closed. This request must be fed back to the binary input *Line Contactor closed*. This is done through a direct connection or by using a delay block to simulate in more detail.

Dialog box



The "Infeed_Tel_370" dialog box shows the current status of the control and status words in telegram 370.

SiemensMomentumReduction – Analysis of the torque reduction value from a PROFIdrive telegram

Symbol



Function

The *SiemensMomentumReduction* component type is used to enhance a PROFIdrive telegram according to the PROFIdrive profile (Technical Specification, Version 4.2, Edition: October 2015 (Order No.: 3.172)) by the manufacturer-specific functionality for the transmission of a torque reduction value.

	PZD 1	PZD 2	PZD 3	PZD 4	PZD 5	PZD 6	PZD 7	PZD 8	PZD 9
TEL_5 Nominal	STW 1	NSOLL_B		STW 2	G1_STW	XERR		KPC	
TEL_5 Actual	ZSW 1	NIST_B		ZSW 2	G1_ZSW	G1_XIST 1		G1_XIST 2	

The *SiemensMomentumReduction* component type denormalizes the transferred torque reduction specification.

Connectors

Input	Description
MOMRED	Standardized torque reduction specification from the PROFIdrive telegram
DriveIn	Structure for docking to upstream PROFIdrive components

Output	Description
MELDW	Currently not used and is always 0
M_RED	Reduced torque with reference to Torque_RedScale parameter
M_Lower_Limit	Reduced low torque limit
M_Upper_Limit	Reduced high torque limit
DriveOut	Structure for connecting additional PROFIdrive components

Parameter

Parameter	Description
Torque_Nominal	Here you enter the reference torque.
Torque_UpperLimit	Here you enter the top physical speed limit of your drive.
Torque_LowerLimit	Here you enter the low physical speed limit of your drive.
Torque_RedScale	Here you specify the percentage to which the torque reduction is to be normalized. The reference torque is typically 100%.

Sinamics – SINAMICS frequency converter

Symbol



Function

The specific additional functions for the SINAMICS frequency converter are implemented in the *Sinamics* component type. The implementation is coordinated with the processing stages contained in the SINA_GS function block. The SINA_GS function block is located in the function block library DriveES-PCS 7 .

The *Sinamics* component type can only be used in combination with the *PROFIdrive1* or *PROFIdrive2* component type.

Table 8-14 Sinamics-specific evaluation of the control word

STW1.11 **Inversion of setpoint value**	**NSOLL**
0	NSOLL
1	–NSOLL

The speed actual value *NIst* and the bit 11 of the status word are set according to the relationships listed in the table below.

Table 8-15 Sinamics-specific states

NIst	**ZSW1.11** **Positive direction of rotation**
NIst >= 0	1
NIst < 0	0

DCMaster – SIMOREG DC Master power converter

Symbol



Function

The specific additional functions for the SIMOREG DC Master power converter are implemented in the *DCMaster* component type. The implementation is adapted to the processing steps contained in the SIMO_DC function block from the Drive ES PCS 7 function block library.

The *DCMaster* component type can only be used in combination with the *PROFIdrive1* or *PROFIdrive2* component type.

Table 8-16 DCMaster-specific evaluation of the control word

STW1.11 Enable positive direction of rotation	STW1.12 Enable negative direction of rotation	NSOLL
0	0	0
0	1	-|NSOLL|
1	0	|NSOLL|
1	1	NSOLL

The speed actual value *NIst* and the bits 12 and 14 of the status word are set according to the relationships listed in the table below.

Table 8-17 DCMaster-specific states

NIst	ZSW1.12 Line contactor requirement	ZSW1.14 Positive direction of rotation
NIst > 0	1	1
NIst = 0	0	1
NIst < 0	1	0

Masterdrive – SIMOVERT Masterdrive frequency converter

Symbol



Function

The specific additional functions for the SIMOVERT Masterdrive frequency converter are implemented in the *Masterdrive* component type. The implementation is coordinated with the processing stages contained in the SIMO_MD function block. The SIMO_MD function block is located in the function block library DriveES-PCS7.

The *Masterdrive* component type can only be used in combination with the *PROFIdrive1* or *PROFIdrive2* component type.

Table 8-18 Masterdrive-specific evaluation of the control word

STW1.11 Enable positive direction of rotation	STW1.12 Enable negative direction of rotation	NSOLL
0	0	0
0	1	-|NSOLL|
1	0	|NSOLL|
1	1	NSOLL

The speed actual value *NIst* and the bits 12 and 14 of the status word are set according to the relationships listed in the table below.

Table 8-19 Masterdrive-specific states

NIst	ZSW1.12 Line contactor requirement	ZSW1.14 Positive direction of rotation
NIst > 0	1	1
NIst = 0	0	1
NIst < 0	1	0

Micromaster3 – MICROMASTER Type 3 frequency converter

Symbol



Function

The specific additional functions for the MICROMASTER Type 3 frequency converter are implemented in the *Micromaster3* component type. The implementation is coordinated with the processing stages contained in the SIMO_MM3 function block. The SIMO_MM3 function block is located in the function block library DriveES-PCS7.

The *Micromaster3* component type can only be used in combination with the *PROFIdrive1* or *PROFIdrive2* component type.

Table 8-20 Micromaster3-specific evaluation of the control word

STW1.14 **Clockwise rotation**	**NSOLL**
0	- NSOLL
1	NSOLL

The speed actual value *NIst* and the bit 14 of the status word are set according to the relationships listed in the table below.

Table 8-21 Micromaster3-specific states

NIst	**ZSW1.14** **Clockwise rotation**
NIST >= 0	1
NIST < 0	0

Micromaster4 – MICROMASTER Type 4 frequency converter

Symbol



Function

The specific additional functions for the MICROMASTER Type 4 frequency converter are implemented in the *Micromaster4* component type. The implementation is coordinated with the processing stages contained in the SIMO_MM4 function block. The SIMO_MM4 function block is located in the function block library DriveES-PCS7.

The *Micromaster4* component type can only be used in combination with the *PROFIdrive1* or *PROFIdrive2* component type.

Table 8-22 Micromaster4-specific evaluation of the control word

STW1.11 **Inversion of setpoint value**	**NSOLL**
0	NSOLL
1	- NSOLL

The speed actual value *NIst* and the bit 11 of the status word are set according to the relationships listed in the table below.

Table 8-23 Micromaster4-specific states

NIst	**ZSW1.11** **Positive direction of rotation**
NIST >= 0	1
NIST < 0	0

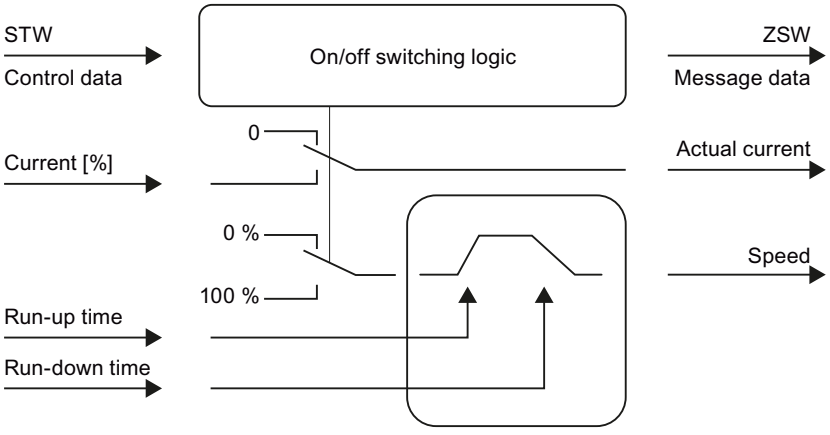
8.1.4.6 SIMOCODE pro motor control devices

SIMOCODE pro motor management and control devices are used to switch motors on and off and to monitor the resulting currents. As an option, SIMOCODE pro can also be used to record other measured values and to access comprehensive statistical evaluations. A SIMOCODE pro device is connected to the controller as an individual PROFIBUS DP slave.

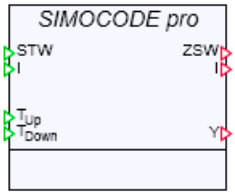
By configuring the device accordingly, a SIMOCODE pro can be employed for very different tasks. For example, it can operate as direct-on-line starter, reversing starter, star-delta starter with or without reversal of rotation, pole switching device with or without reversal of rotation, as well as positioner, solenoid valve drive, overload relay or power circuit breaker. The directory *SIMOCODEpro* in the Drives Library contains component types that emulate the various control functions of SIMOCODE pro. These component types form the SIMOCODEpro Library of SIMIT.

Basic functions of SIMOCODE pro components

Each component type of the SIMOCODEpro Library contains, as a basic function, a simple emulation of the motor together with the logic for switching the motor on and off.



The corresponding connectors of the *SIMOCODEpro* component types can be seen in the symbols depicted in the figure below.

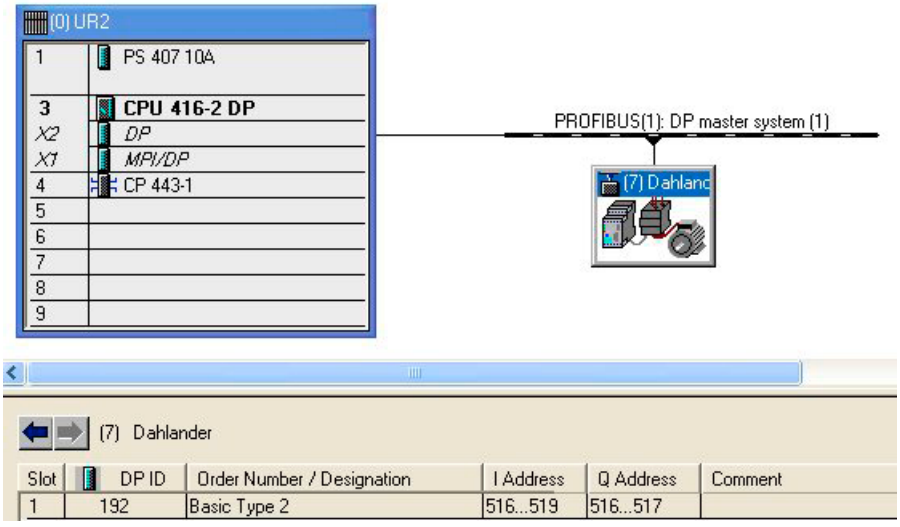


All SIMOCODE pro devices are accessed via control data and return their current status to the controller by means of message data. The content of the control and message data depends on which control function is implemented. Only the cyclical control and message data is processed by the *SIMOCODEpro* component types; acyclical data such as statistical data, for example, are not taken into account. The components can interface with the controller in the following ways:

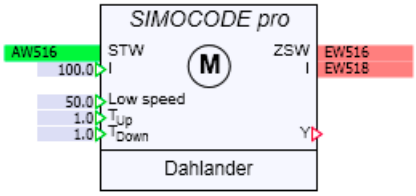
- The control data word (2 bytes) of the controller outputs is connected to the integer input *STW* of the component.
- The first data word of the message data in the controller inputs contains the binary feedback and is connected to the integer output *ZSW*. Analog message data, for example, the actual current value, is transferred to the controller in the second data word.

This configuration corresponds to SIMOCODE pro C or, in the case of SIMOCODE pro V, to basic type 2.

In the figure below, an example of a SIMATIC configuration for a Dahlander motor is shown.



The figure below shows the *SIMOCODEpro* component and the associated process data.



The actual current *I* is communicated to the controller as a percentage value of the current setting (rated motor current). The current value at input *I* is interconnected as the actual current

to the output *I*, provided that the motor is switched on. The input has a default value of 100%, which means the actual current equals the current setting. The switching on and off of the motor is controlled by a ramp function.

The relationship between current and motor load is not taken into account. This can however easily be added in the simulation, for example, by not setting the current as a constant value but by using suitable functions to make it dependent on the motor speed or on the process instead. You can find additional information on this in section: Individual adaptations (Page 556).

The speed of the motor is available as a percentage value at output *Y* of a *SIMOCODEpro* component. The run-up and run-down times of the motor are set on the two analog inputs T_{Up} and T_{Down}. T_{Up} is the time in seconds it takes the motor to run up from standstill to the nominal speed; T_{Down} is the time in seconds it takes for the motor to run down to a standstill from the nominal speed. Both times have a default value of one second. If one of the two input values is negative during the simulation, the message "*x: run-up or run-down time invalid value*" (message category *ERROR*) is generated.

Note

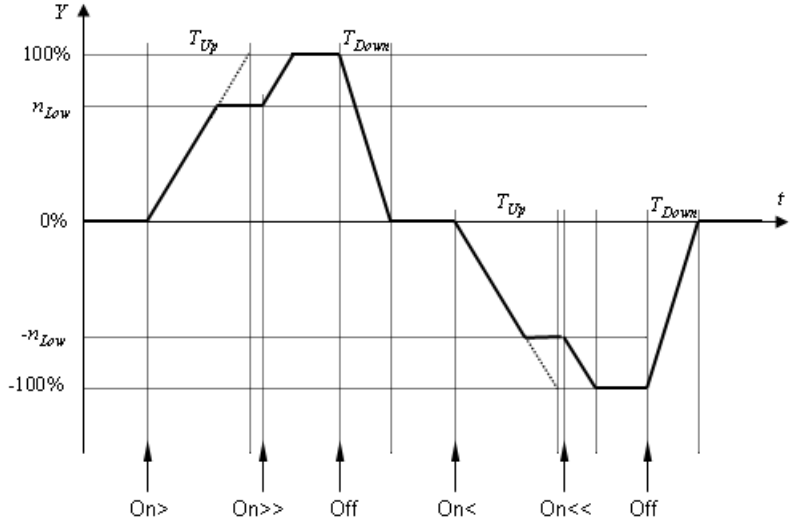
In the case of positioners and solenoid valves, the run-up and run-down times mean the opening and closing time T_{Up} and T_{Down} of the valve.

The speed of the motor is not specified by the controller nor detected by the SIMOCODE pro devices. The speed value is therefore not communicated to the controller. Its purpose is to provide another source of motor speed input for the simulation. Consequently, the run-up and run-down times are only of minor importance.

Switchover pauses and interlock times are ignored by the *SIMOCODEpro* component types. Corresponding feedback for the controller is not generated.

Ramp function

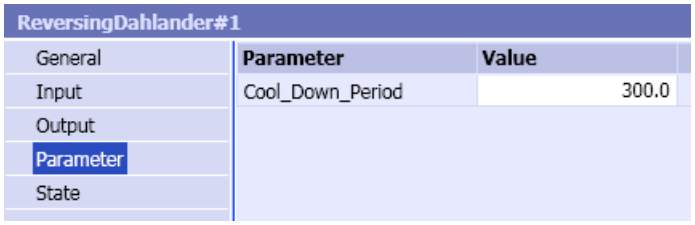
The motor speed is formed using a ramp function. The most general form of this ramp for a drive with two speeds in two directions of rotation is illustrated in the figure below.



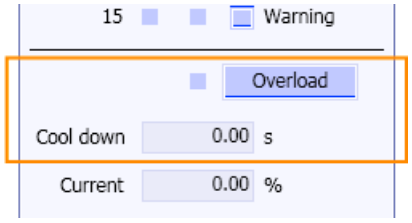
The increase and decrease times of the ramp correspond to the run-up and run-down times T_{Up} and T_{Down} of the motors. Depending on how SIMOCODE pro is configured, the motor speed *Y* is generated as a percentage value in the range –100% to 100%.

Overload behavior

Overload can be set using the *Overload* switch in the operating window of a component. The motor is switched off in the event of an overload. A thermal switch-on inhibitor determines when the motor can be switched on again (cooling down time). The cooling down time begins when the overload no longer exists. The cooling down time is set in the *Cool_Down_Period* parameter and has a default value of 300 seconds, as shown in the figure below.



As shown in the figure below, the remaining cooling down time is displayed in the *Cool down* field of the operating window:



During this time the drive is disabled, which means it cannot be restarted.

The thermal switch-on inhibitor is reset by setting the emergency start (*EM-Start*, *STW.12*) bit. The drive can be restarted immediately.

The advance warning for overload (*I >115%*, *ZSW.11*) is set as soon as the input current exceeds 115%.

Standard assignments in the control and message data

The functions shown in the table below are used as standard for bits 11 to 15 of the control word (STW. 11 to STW.15) in all *SIMOCODEpro* component types. The Emergency Start function does not apply to solenoid valves.

Table 8-24 Standard assignments in the control word

Name	Control data	
Test1	Test function: reset after 5 seconds	STW.11
EM-Start	Emergency start	STW.12
Remote	Operating mode switch S1	STW.13
Reset	Reset device	STW.14

The test function *Test1* resets the SIMOCODE pro device five seconds after setting the signal. The motor is switched off.

Setting *EM-Start* resets the thermal switch-on inhibitor if it has been triggered by an overload. The motor can then be switched on again immediately.

The *Remote* command displays the controller default for operating mode switch S1. This command has no functional effect on the simulation.

The SIMOCODE pro device is reset using the *Reset* command. This switches the motor off.

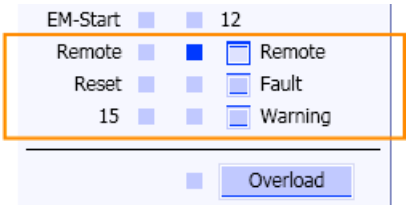
The table below provides an overview of the standard message data for all *SIMOCODEpro* component types. The overload advance warning does not apply to solenoid valves.

Table 8-25 Standard assignments in the status word

Name	Message data	
I>115%	Overload advance warning	ZSW.11
Remote	Remote operating mode	ZSW.13
Fault	General fault	ZSW.14
Warning	General warning	ZSW.15

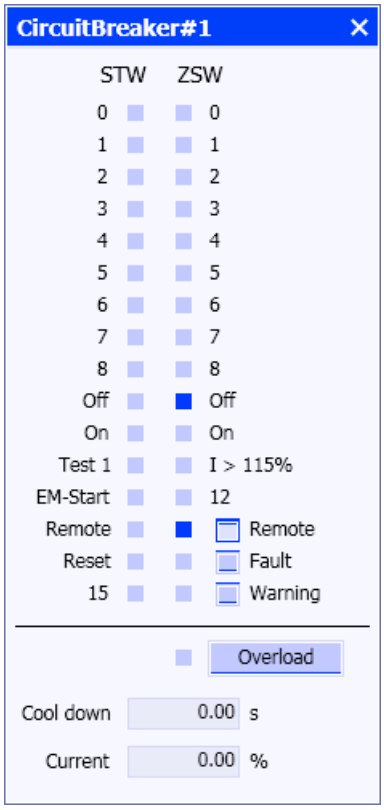
If the current value rises above 115%, bit ZSW.11 is set as advance warning of an overload.

The *Remote*, *Fault* and *Warning* signals can be set in the operating window of the component. The *Remote* signal has a default value of one, which means the switch is closed.



Operating window of the SIMOCODEpro components

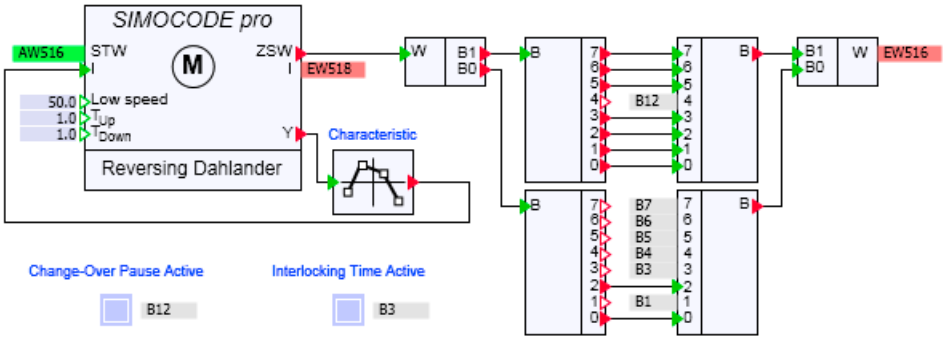
All *SIMOCODEpro* components have an operating window in which the signals of the control word and status word are displayed. The names of the control word signals used by each component are shown in the operating window. The same applies to the message data signals that are manipulated by the components. Signals indicated by numbers have no function in the component and are only displayed in the operating window. The figure below shows the operating window for the component of type *ReversingDahlander*. The operating windows for components of other types are designed accordingly.



Individual adaptations

Depending on how parameters have been assigned to a SIMOCODE pro device, signals in the control and message data can have meanings that are not present in the SIMOCODEpro Library components. However, these signals can be included in the simulation by a simple connection to other components from the SIMIT Basic Library.

In the example of a *ReversingDahlander* component illustrated in the figure below, the interlocking time and the change-over pause can be set manually. Neither signal is set in the component. Here, the status word on the output of the component has been divided into its individual signals by the conversion components (*Word2Byte*, *Byte2Bit*). The signals set in the component are converted into a word again by the signals linked via the global connectors, and communicated to the *IW516* signal input of the automation system.



An example is also shown of how the current on the input of the *ReversingDahlander* component can be mapped from the speed of the motor on output *Y* of the *ReversingDahlander* component using a suitable trend function (*Characteristic*) and adjusted as required.

Specific SIMOCODE pro devices

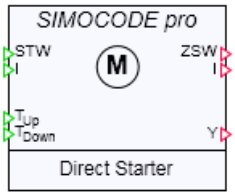
The SIMOCODEpro library contains ten different *SIMOCODEpro* component types. These component types emulate the various control functions of a SIMOCODE pro. Each control function is emulated by a component type. The simulation function of the component types is tailored to the control variants (function blocks) in PCS 7.

Table 8-26 Supported control functions of the SIMOCODE pro

SIMIT component types	Control function	PCS 7 FB
DirectStarter	Direct starter	SMC_DIR
ReversingStarter	Reversing starter	SMC_REV
StarDeltaStarter	Star-delta starter	SMC_STAR
ReversingStarDelta	Reversing star-delta starter	SMC_REVS
Dahlander	Dahlander / pole-changing switch	SMC_DAHL
ReversingDahlander	Dahlander / pole-changing switch with reversal of direction of rotation	SMC_REVD
Valve	Solenoid valve	SMC_VAL
Positioner	Slider	SMC_POS
OverloadRelay	Overload	SMC_OVL
CircuitBreaker	Circuit breaker	SMC_CB

DirectStarter – Direct starter

Symbol



Function

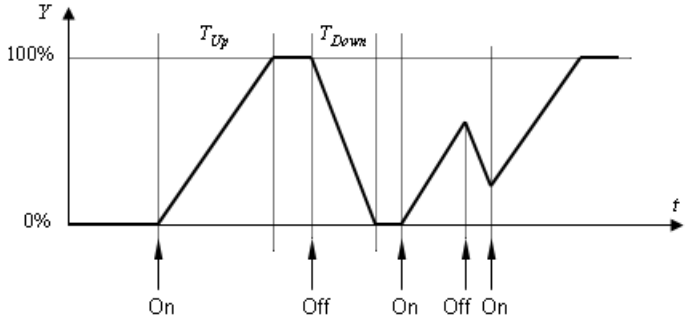
The component type *DirectStarter* simulates drives that have a single direction of rotation and can be switched on and off directly. The table below provides an overview of the relevant control data *STW* and message data *STW* for this application.

Table 8-27 Control and message data of the DirectStarter component

Name	Control data		Message data	
Off	Switch off drive	STW.9	Drive switched off	STW.9
On	Switch on drive	STW.10	Drive switched on	STW.10

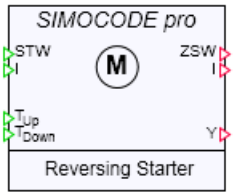
The commands "Switch off drive" (Off, STW.9) and "Reset" (RESET, STW.14) have priority over the command "Switch on drive" (On, STW.10).

The actual current value at output *I* is set to the value at the current input *I* as soon as the drive is switched on. If the drive is switched off, it is set to zero. The speed value at the output *Y* is formed as a percentage value with the help of the ramp function: $0 \leq Y \leq 100$.



ReversingStarter – Reversing starter

Symbol



Function

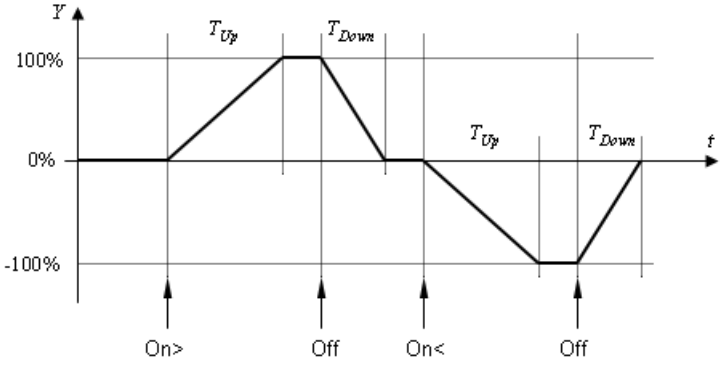
The *ReversingStarter* component type simulates drives with two directions of rotation, which can be switched on and off directly in both directions. The table below provides an overview of the relevant control data *STW* and message data *ZSW* for this application.

Table 8-28 Control and message data of the DirectStarter component

Name	Control data		Message data	
On<	Switch on counterclockwise rotation	STW.8	Counterclockwise rotation switched on	ZSW.8
Off	Switch off drive	STW.9	Drive switched off	ZSW.9
On>	Switch on clockwise rotation	STW.10	Clockwise rotation switched on	ZSW.10

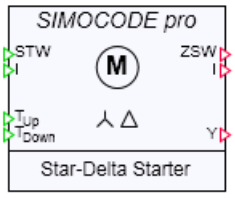
The commands "Switch off drive" (Off, STW.9) and "Reset" (RESET, STW.14) have priority over the command "Switch on counterclockwise rotation" or "Switch on clockwise rotation" (On<, STW.8 or On>, STW.10). Simultaneous switch-on commands for both directions of rotation do not alter the status of the drive; they are ignored.

The actual current value at output *I* is set to the value at the current input *I* as soon as the drive is switched on in one of the two directions of rotation. If the drive is switched off, the actual current value is set to zero. The speed value at the output *y* is formed as a percentage value with the help of the ramp function: $-100 \leq Y \leq 100$.



StarDeltaStarter – Star-delta starter

Symbol



Function

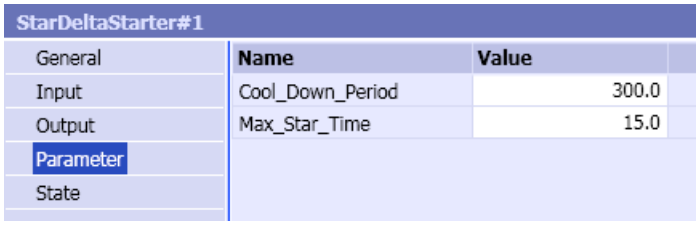
The *StarDeltaStarter* component type simulates drives with star-delta switchover. The table below provides an overview of the relevant control data *STW* and message data *STW* for this application.

Table 8-29 Control and message data of the StarDeltaStarter component

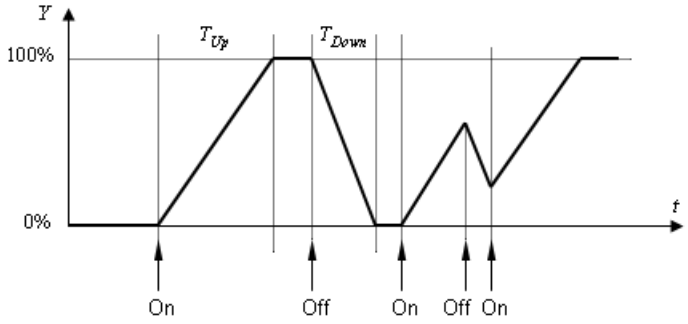
Name	Control data		Message data	
Off	Switch off drive	STW.9	Drive switched off	STW.9
On	Switch on drive	STW.10	Delta mode switched on	STW.10

The commands "Switch off drive" (Off, STW.9) and "Reset" (RESET, STW.14) have priority over the command "Switch on drive" (On, STW.10).

The maximum time for star mode is set in the parameter *Max_Star_Time*. The default time is 15 seconds. The switchover to delta mode (STW.10) is performed when the maximum star mode time has been reached or if a value of less than 90% is present on the current input.

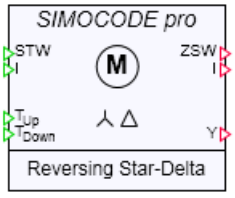


The actual current value at output *I* is set to the value at the current input *I* as soon as the drive is switched on. If the drive is switched off, the actual current value is set to zero. The speed value at the output *Y* is formed as a percentage value with the help of the ramp function: $0 \leq Y \leq 100$.



ReversingStarDelta – Star-delta starter with reversal of direction of rotation

Symbol



Function

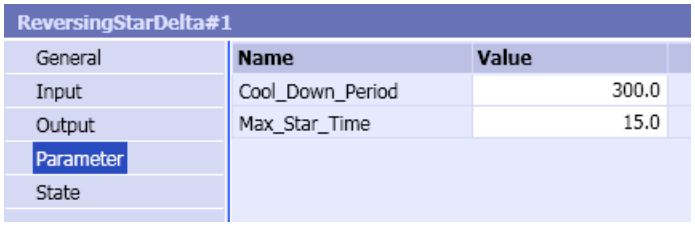
The *ReversingStarDelta* component type simulates drives with star-delta switchover in both directions of rotation. The table below provides an overview of the relevant control data *STW* and message data *ZSW* for this application.

Table 8-30 Control and message data for the ReversingStarDelta component

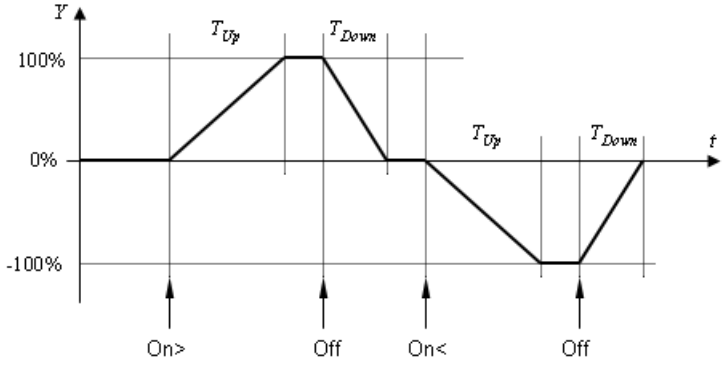
Name	Control data		Message data	
On<	Switch on counterclockwise rotation	STW.8	Counterclockwise delta mode switched on	ZSW.8
Off	Switch off drive	STW.9	Drive switched off	ZSW.9
On>	Switch on clockwise rotation	STW.10	Clockwise delta mode switched on	ZSW.10

The commands "Switch off drive" (Off, STW.9) and "Reset" (RESET, STW.14) have priority over the command "Switch on counterclockwise rotation" or "Switch on clockwise rotation" (On<, STW.8 or On>, STW.10). Simultaneous switch-on commands for both directions of rotation do not alter the status of the drive; they are ignored.

The maximum time for star mode is set in the parameter *Max_Star_Time*. The default time is 15 seconds. The direction-dependent switchover to delta mode (ZSW.10 or ZSW.8) is performed when the maximum star mode time has been reached or if a value of less than 90% is present at the current input.

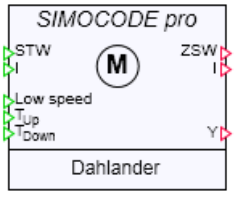


The actual current value at output *I* is set to the value at the current input *I* as soon as the drive is switched on in one of the two directions of rotation. If the drive is switched off, the actual current value is set to zero. The speed value at the output *y* is formed as a percentage value with the help of the ramp function: –100 ≤ *Y* ≤ 100.



Dahlander – Dahlander starter or pole-changing switch

Symbol



Function

The *Dahlander* component type simulates drives with a single direction of rotation and two speeds: full speed and low speed. The table below provides an overview of the relevant control data *STW* and message data *ZSW* for this application.

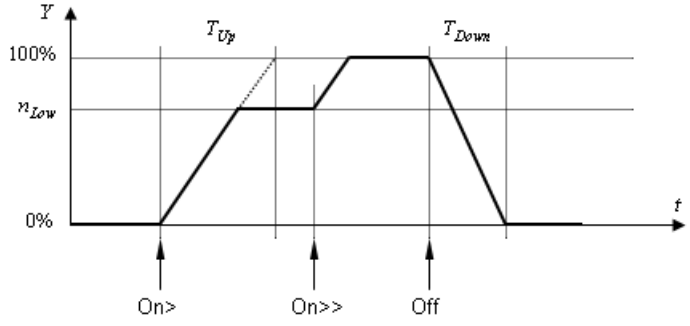
Table 8-31 Control and message data of the Dahlander component

Name	Control data		Message data	
On>>	Switch drive to full speed	STW.8	Drive switched to full speed	ZSW.8
Off	Switch off drive	STW.9	Drive switched off	ZSW.9
On>	Switch drive to low speed	STW.10	Drive switched to low speed	ZSW.10

The commands "Switch off drive" (Off, STW.9) and "Reset" (RESET, STW.14) have priority over the commands "Switch to full speed" (On>>, STW.8) or "Switch to low speed" (On>, STW.10). Simultaneous switch-on commands for both speeds do not alter the status of the drive; they are ignored.

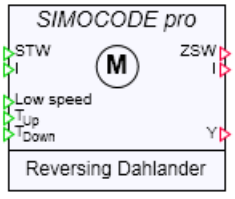
The actual current value at output *I* is set to the value at the current input *I* as soon as the drive is switched to one of the two speeds. If the drive is switched off, the actual current value is set to zero.

The low speed n_{Low} is set as a percentage value on the input *Low speed*. The default low speed value is 50%. The speed value at the output *Y* is formed as a percentage value with the help of the ramp function: $0 \leq Y \leq 100$.



ReversingDahlander – Dahlander starter or pole-changing switch with reversal of direction of rotation

Symbol



Function

The *ReversingDahlander* component type simulates drives with two speeds of rotation in two directions. The table below provides an overview of the relevant control data *STW* and message data *ZSW* for this application.

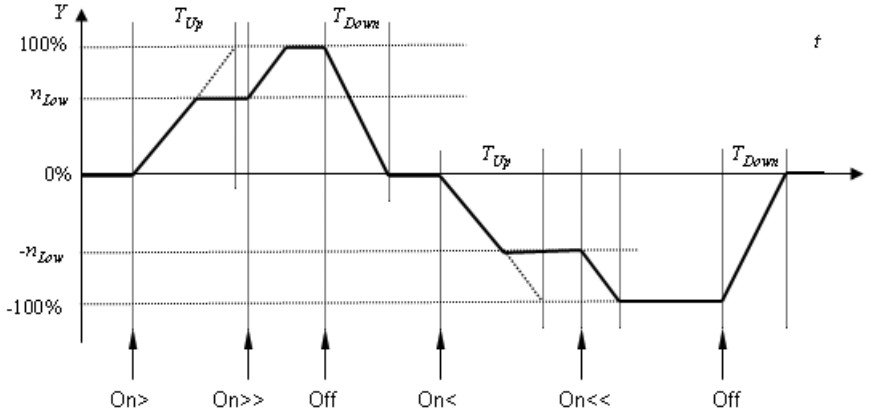
Table 8-32 Control and message data for the ReversingDahlander component

Name	Control data		Message data	
On<<	Switch drive to full speed counterclockwise	STW.0	Drive switched to full speed counterclockwise	ZSW.0
On<	Switch drive to low speed counterclockwise	STW.2	Drive switched to low speed counterclockwise	ZSW.2
On>>	Switch drive to full speed clockwise	STW.8	Drive switched to full speed clockwise	ZSW.8
On>	Switch drive to low speed clockwise	STW.10	Drive switched to low speed clockwise	ZSW.10
Off	Switch off drive	STW.9	Drive switched off	ZSW.9

The commands "Switch off drive" (Off, STW.9) and "Reset" (RESET, STW.14) have priority over the switch-on commands "Full speed counterclockwise" (On<<, STW.0), "Low speed counterclockwise" (On<, STW.2), "Full speed clockwise" (On>>, STW. 8) and "Low speed clockwise" (On >, STW.10). Several simultaneous switch-on commands do not alter the status of the drive; they are ignored.

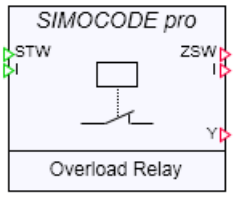
The actual current value at output *I* is set to the value at the current input *I* as soon as the drive is switched on. If the drive is switched off, the actual current value is set to zero.

The low speed n_{Low} is set as a percentage value on the input *Low speed*. The default low speed value is 50%. The speed value at the output *Y* is formed as a percentage value with the help of the ramp function: $-100 \leq Y \leq 100$.



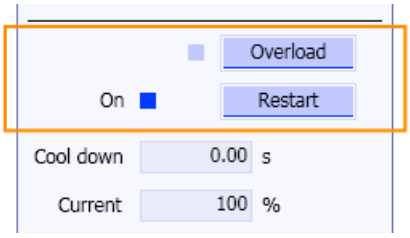
OverloadRelay – Overload relay

Symbol

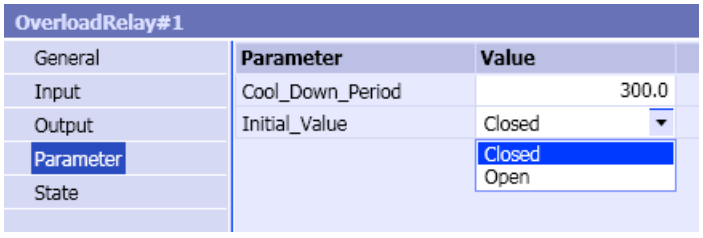


Function

The *OverloadRelay* component type simulates drives with overload monitoring. The *Overload* command triggers an overload, which means the drive is switched off. The *Restart* command switches the drive on. The commands can only be set in the operating window of the component.



The drive can be initialized as switched on or switched off using the *Initial_Value* parameter: switched on with the value *Closed*, switched off with the value *Open*. The default setting for *Initial_Value* is *Closed*.

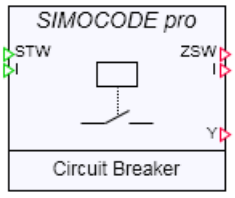


The actual current value at output *I* is set to the value at the current input *I* as soon as the drive is switched on. If the drive is switched off, the actual current value is set to zero.

The speed value on the output *Y* is set to one hundred if the drive is on and to zero if the drive is off.

CircuitBreaker – Circuit breaker

Symbol



Function

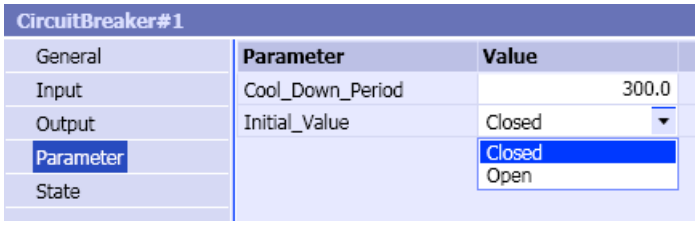
The *CircuitBreaker* component type simulates drives with switching characteristics. The table below provides an overview of the relevant control data *STW* and message data *ZSW* for this application.

Table 8-33 Control and message data of the CircuitBreaker component

Name	Control data		Message data	
Off	Open switch	STW.9	Switch opening / open	ZSW.9
On	Close switch	STW.10	Switch closing / closed	ZSW.10

The commands "Open switch" (Off, STW.9) and "Reset" (RESET, STW.14) have priority over the closing command (On, STW.10).

The switch can be initialized as closed or open using the *Initial_Value* parameter: closed with the value *Closed*, opened with the value *Open*. The default setting for *Initial_Value* is *Closed*.

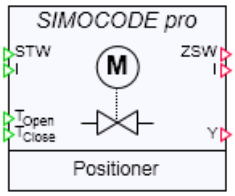


The actual current value at output *I* is set to the value at the current input *I* as soon as the switch closes. If the switch is opened, the actual current value is set to zero.

The speed value on the output *Y* is set to one hundred if the drive is on and to zero if the drive is off.

Positioner – Slide valve/Positioner

Symbol



Function

The *Positioner* component type simulates positioning drives for slide valves, adjusting valves, etc. The table below provides an overview of the relevant control data *STW* and message data *ZSW* for this application.

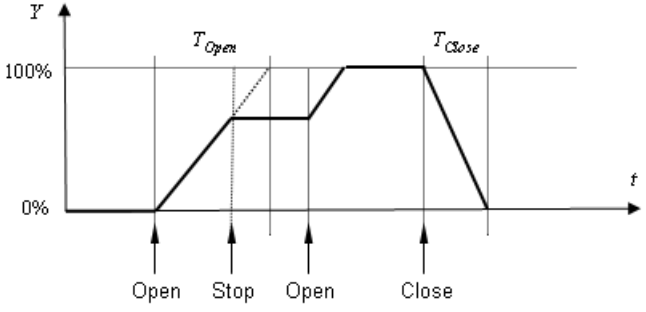
The commands "Switch off drive" (Off, STW.9) and "Reset" (RESET, STW.14) have priority over the opening and closing commands (Open, STW.10 and Close, STW.8). Simultaneously set opening and closing commands do not alter the status of the drive; they are ignored.

Table 8-34 Control and message data of the Positioner component

Name	Control data		Message data	
Close	Close slide valve	STW.8	Slide valve closed	ZSW.8
			Slide valve closes	ZSW.2
Stop	Stop slide valve	STW.9	Slide valve stops	ZSW.9
Open	Open slide valve	STW.10	Slide valve open	ZSW.10
			Slide valve opens	ZSW.0

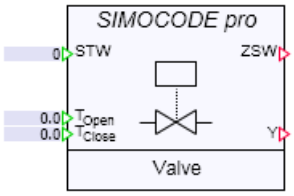
The actual current value at the output *I* is set to the value at the current input "*I* as long as the slide valve is opening (ZSW.0) or closing (ZSW.2). The actual current value is set to zero if the slide valve is stationary and when it is closed or open.

The position value at the output *Y* is formed as a percentage value with the help of a ramp function: $0 \leq Y \leq 100$. The value zero represents a closed slide valve, the value one hundred an opened slide valve. The default opening and closing times of the slide valves are both set to one second.



Valve – Solenoid valve

Symbol



Function

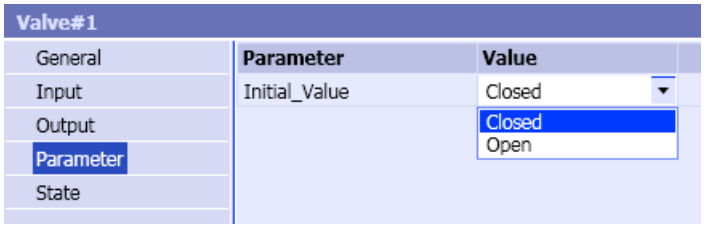
The *Valve* component type simulates drives for solenoid valves. The table below provides an overview of the relevant control data *STW* and message data *STW* for this application.

Table 8-35 Control and message data of the Valve component

Name	Control data		Message data	
Close	Close valve	STW.9	Valve closing / closed	STW.9
Open	Open valve	STW.10	Valve opening / opened	STW.10

The commands "Close valve" (Close, STW.9) and "Reset" (RESET, STW.14) have priority over the opening command (Open, STW.10 and Close, STW.8).

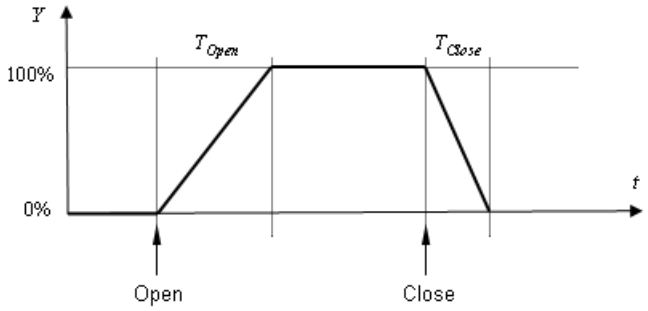
The valve can be initialized as closed or open using the *Initial_Value* parameter: closed with the value *Closed*, opened with the value *Open*. The default setting for *Initial_Value* is *Closed*.



The setting value at the output *Y* is formed as a percentage value with the help of a ramp function.

$0 \leq Y \leq 100$

The value zero represents a closed valve, the value one hundred an opened valve.



The default opening and closing times of the solenoid valve are zero, which means the opening and closing of the valve is done with a high gradient.

The *Valve* component does not have a current monitoring function and therefore no overload behavior, neither does it have a current input nor a current output.

8.1.5 Communication components

8.1.5.1 Components for SIMATIC

Multi HART

Symbol

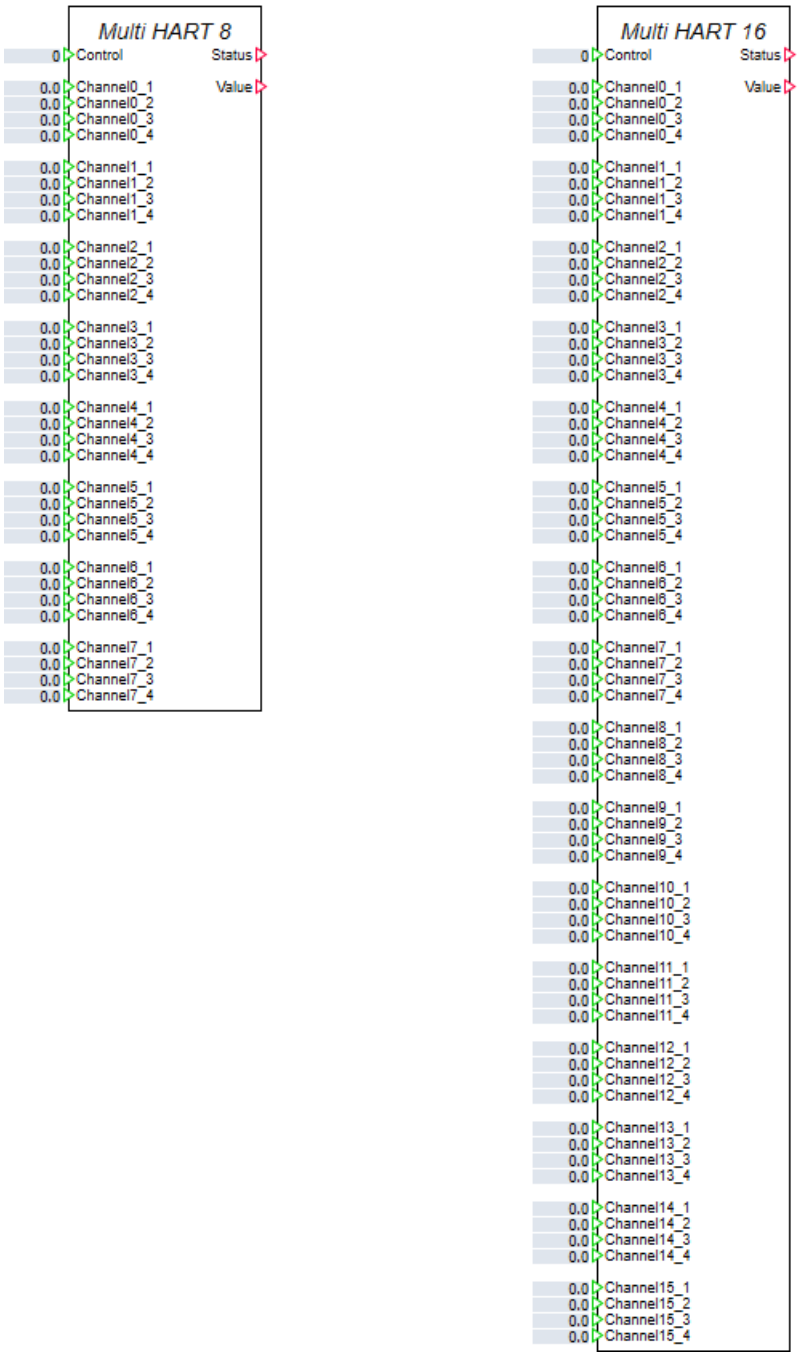


Figure 8-1 Symbol for the Multi HART 8 component type (left) and the Multi HART 16 component type (right)

Function

The *Multi HART 8* and *Multi HART 16* component types are multiplexers for the ET200SP HA hardware family.

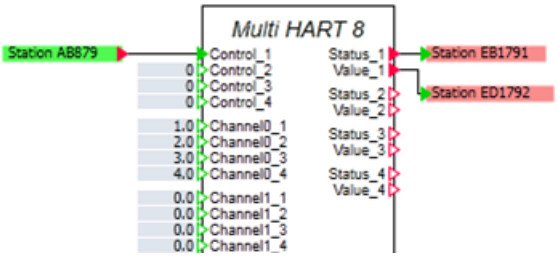
Depending on the value of the selection input Control, the component type *Multi HART 8* / *Multi HART 16* switches one of the 32 / 64 inputs to the Value output. The Status output receives the same value as the Control selection input.

The default value of the selection input Control is 0. All other inputs are set to 0.0 by default.

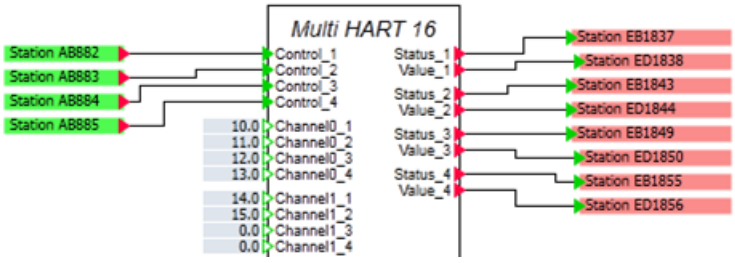
Parameter assignment

The configuration of the HART secondary variables in ET 200SP HA defines how many selection inputs you need to connect in SIMIT:

- Configuration as "1 multiHART":



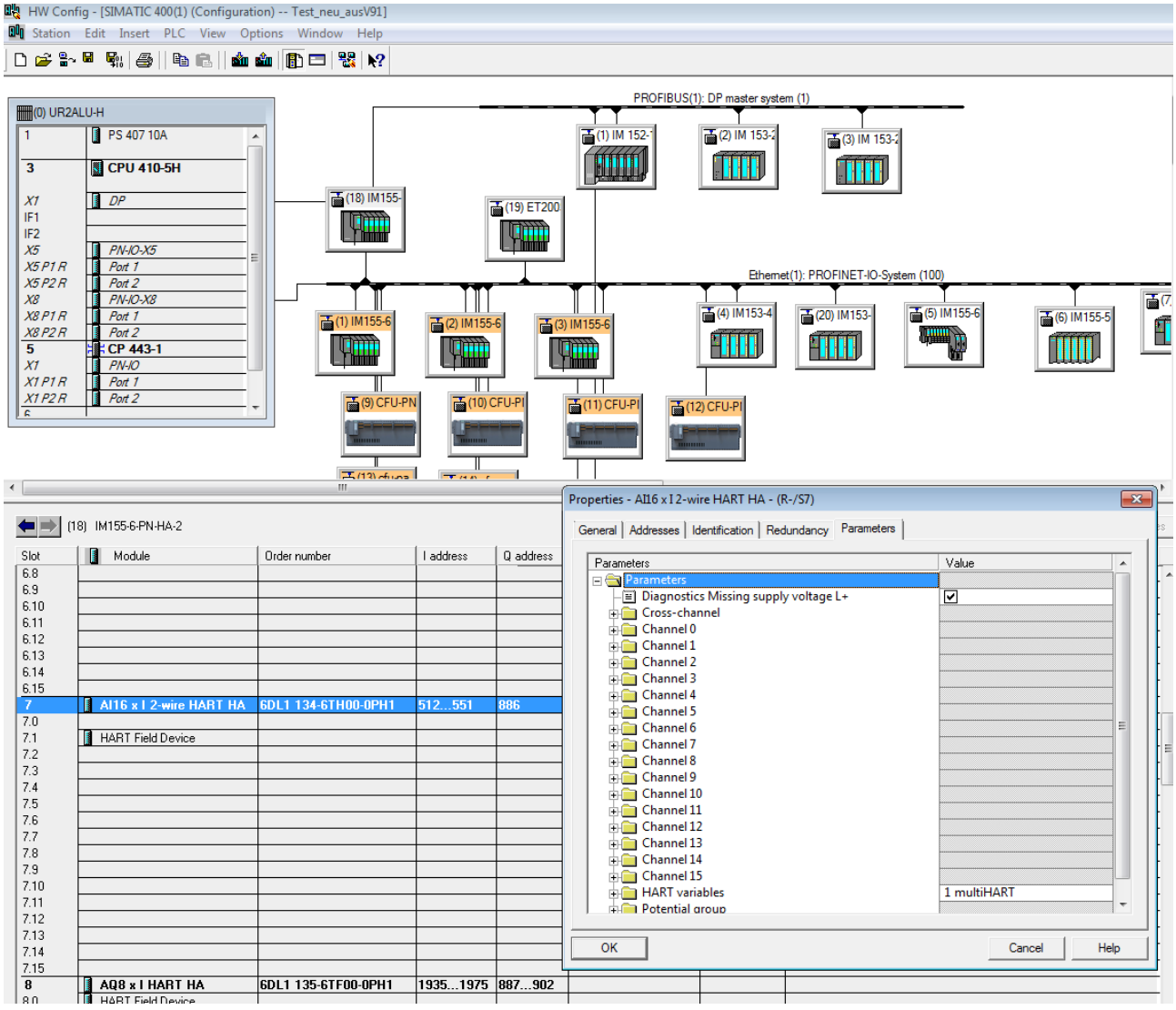
- Configuration as "4 multiHART":



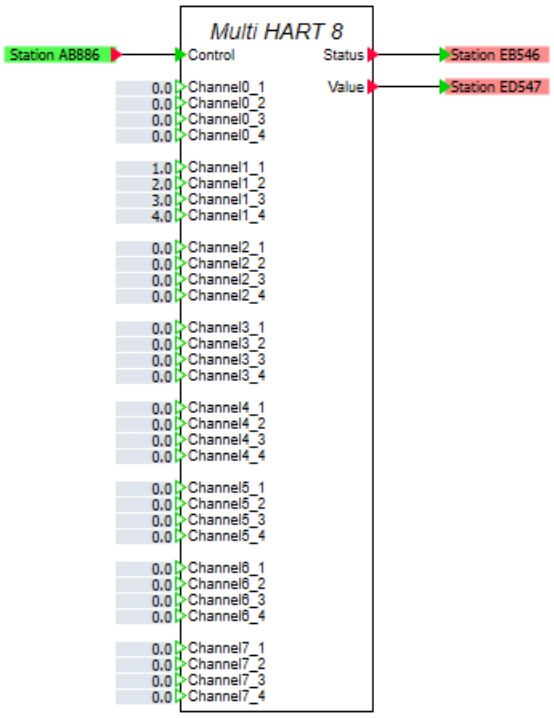
Example

The Multi HART component is used for the simulation of HART secondary variables when the HART secondary variables in ET200SP HA are configured as "1 multiHART" or "4 multiHART" and the Pcs7HaAI block is used in the PCS 7 project. The Multi HART component is not required for "8 HART" configuration, since the secondary variables are available directly via the process image.

In the following example, the four HART secondary variables are read from channel 2 of module 7.



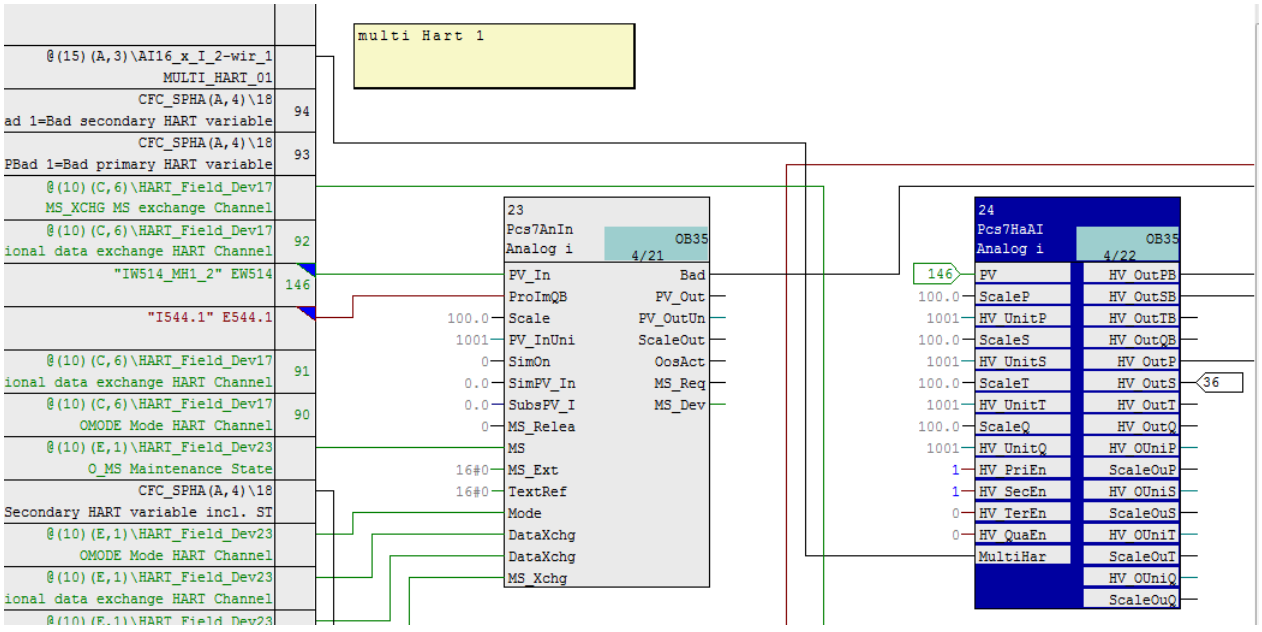
As described in the help for ET200SP HA, the control byte for AB886 and the status byte for EB546 is included in this configuration. AB886 is linked to the Control input of the Multi HART component and determines the HART secondary variable to be currently read. The HART secondary variable just read is passed to ED547.



In the example, control is generated by Pcs7HaAI and the following sequence is executed:

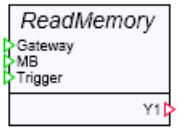
Control = Channel1_1. Then Status = Channel1_1 and Value = 1.0.
Control = Channel1_2. Then Status = Channel1_2 and Value = 2.0.
Control = Channel1_3. Then Status = Channel1_3 and Value = 3.0.
Control = Channel1_4. Then Status = Channel1_4 and Value = 4.0.

The configuration in CFC looks like this:



ReadMemory – Reading a bit memory address area

Symbol



Function

The *ReadMemory* component type enables one or more successive bytes from the bit memory address area of a controller to be read. Enter the address of the first byte to be read at the input *MB*. The read operation is executed when a rising edge occurs at the *Trigger* input, which means a change from False to True.

The number *N* of outputs can be varied. It can be changed by "graphically scaling" the component on a chart. You can specify a maximum of 32 outputs, which means you can read a maximum of 32 bytes with a component of this type. The bytes that are read are available at the outputs *Y1* to *YN*.

Exactly one read operation is triggered while the simulation is being initialized. This means initial values from the bit memory address area are available after initialization even though no trigger signals have been sent.

The component can be operated with the following couplings:

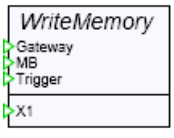
- PLCSIM
- PRODAVE
- Virtual Controller

See also

Access to a data record or memory area (Page 216)

WriteMemory – Writing to a bit memory address area

Symbol



Function

The *WriteMemory* component type allows you to write to one or more successive bytes in the bit memory address area of a controller. Enter the address of the first byte to be written to at the input *MB*. The write operation is executed when a rising edge occurs at the *Trigger* input, which means a change from False to True.

The number *N* of inputs can be varied. It can be changed by "graphically scaling" the component on a chart. You can specify a maximum of 32 inputs, which means you can write to a maximum of 32 bytes in the bit memory address area with a component of this type. The bytes to be written are to be provided at inputs *X1* to *XN*.

Exactly one write operation is triggered while the simulation is being initialized. Initial values can therefore be written to the bit memory address area during the initialization process, even though no trigger signals have been sent.

The component can be operated with the following couplings:

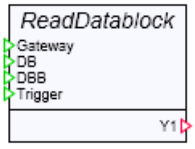
- PLCSIM
- PRODAVE
- Virtual Controller

See also

Access to a data record or memory area (Page 216)

ReadDatablock – Reading a data block

Symbol



Function

The *ReadDatablock* component type enables one or more successive bytes from a data block of a controller to be read. Enter the data block number at the *DB* input and the address of the first byte to be read at the *DBB* input. The read operation is executed when a rising edge occurs at the *Trigger* input, which means a change from False to True.

The number *N* of outputs can be varied. It can be changed by "graphically scaling" the component on a chart. You can specify a maximum of 32 outputs, which means you can read a maximum of 32 bytes with a component of this type. The bytes that are read are available at the outputs *Y1* to *YN*.

Exactly one read operation is triggered while the simulation is being initialized. Initial values from the data block are available after initialization, even though no trigger signals have been sent.

The component can be operated with the following couplings:

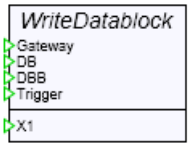
- PLCSIM
- PRODAVE
- Virtual Controller

See also

Access to a data record or memory area (Page 216)

WriteDatablock – Writing to a data block

Symbol



Function

The *WriteDatablock* component allows you to write to one or more successive bytes in the data block of a controller. Enter the data block number at the *DB* input and the address of the first byte to be written at the *DBB* input. The write operation is executed when a rising edge occurs at the *Trigger* input, which means a change from False to True.

The number *N* of inputs can be varied. It can be changed by "graphically scaling" the component on a chart. You can specify a maximum of 32 inputs, which means you can write to a maximum of 32 bytes in the data block with a component of this type. The bytes to be written are to be provided at inputs *X1* to *XN*.

Exactly one write operation is triggered while the simulation is being initialized. Initial values can therefore be written to the data block during the initialization process, even though no trigger signals have been sent.

The component can be operated with the following couplings:

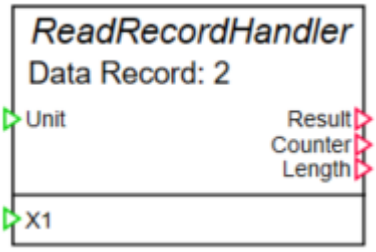
- PLCSIM
- PRODAVE
- Virtual Controller

See also

Access to a data record or memory area (Page 216)

ReadRecordHandler – Response to a read request of the controller

Symbol



Function

The *ReadRecordHandler* component type allows response to a data record request triggered by the real or emulated controller. The *DataRecordNumber* parameter specifies the number of the data record to be answered. A validity check for the data record is not performed.

The *N* number of inputs and thus the number of bytes that are sent with this data record is variable. It can be changed by "graphically scaling" the component on a chart to the maximum 64. The bytes to be transmitted are to be made available at inputs *X1* to *XN*.

The actual length of the last answered data record can be seen at the *Result* output of the component. The actual requested length is available at the *Length* output. As long as the requested length is not greater than the number of *XN* inputs of the component, both values are identical. If a longer data record is requested, the value 0x80B1 (ERROR_LENGTH_INVALID) appears as the *result*.

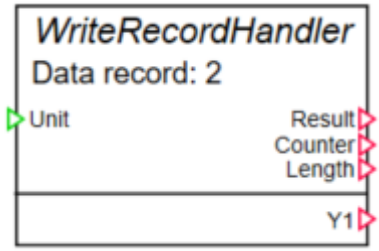
The value of the *Counter output* is increased by one with each response to a data record request.

The component can be operated with the following couplings:

- SIMIT Unit
- SIMIT Virtual Controller
- PLCSIM Advanced

WriteRecordHandler – Response to a write request from the controller

Symbol



Function

The *WriteRecordHandler* component type allows response to a record write request triggered by the real or emulated controller. The *DataRecordNumber* parameter specifies the number of the transferred data record. A validity check for the data record is not performed.

The *N* number of outputs and thus the number of bytes that can be received by this component is variable. It can be changed by "graphically scaling" the component on a chart to the maximum 64. The received bytes are available at the outputs *Y1* to *YN*.

The actual length of the last received data record can be seen at the *Result* output of the component. The actually transferred length is available at the *Length* output. As long as the transferred length is not greater than the number of *YN* outputs of the component, both values are identical. If a longer data record is transferred, the value 0x80B1 (ERROR_LENGTH_INVALID) appears as the *result*.

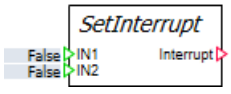
The value of the *Counter output* is increased by one with each response to a data record request.

The component can be operated with the following couplings:

- SIMIT Unit
- SIMIT Virtual Controller
- PLCSIM Advanced

SetInterrupt

Symbol



Function

The component type *SetInterrupt* sets the hardware interrupts in the SIMIT Unit coupling during the simulation.

The number of binary inputs is variable. It can be changed by "graphically scaling" the component on a chart. You can specify a maximum of 64 inputs, which means with a component of this type you can trigger a maximum of 64 hardware interrupts at a hardware object of the SIMIT Unit coupling.

The INX input hereby corresponds to the alarm [X] specified in the coupling. With a rising edge at INX, the corresponding alarm is triggered (incoming), and with a falling edge it is removed again (outgoing). You use the corresponding signal of the hardware object from the SIMIT Unit coupling as interrupt output.

Note

If you do not use this component, direct inputs are no longer possible in the coupling. Direct inputs in the coupling are immediately overwritten by the model values.

See also

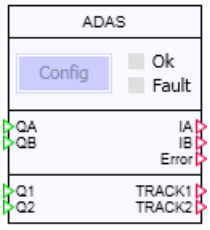
Simulation of alarms (Page 108)

8.1.5.2 Components for SINUMERIK

A component type that enables the transfer of axis values from SINUMERIK to SIMIT can be found in the basic library in the directory *COMMUNICATION* | *SINUMERIK*.

ADAS – AXIS DATA STREAM PER PROFIBUS

Symbol



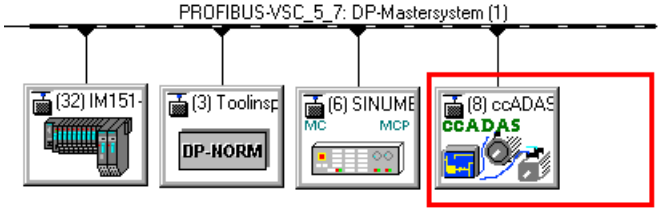
Function

The *ADAS* component type is used for parameter assignment and preparation of axis values, which are transferred to SIMIT via PROFIBUS DP from a SINUMERIK controller using the ADAS software package.

Note

The ADAS software package is not part of the SIMIT product and needs to be purchased separately.

To transfer axis values you need to insert the PROFIBUS slave *ccADAS* into the SINUMERIK hardware configuration.



This slave has 8 bytes each of input and output data for communication and an additional 4 bytes for each axis value.

Profibus

Save and Load

Inputs Reset Filter

Default	Symbol Name	Address	Data Type	System	Slave	Slot	Comment
				1	8		
0		ED17	DWORD	1	8	1	
0		ED21	DWORD	1	8	1	

Outputs Reset Filter

Symbol Name	Address	Data Type	System	Slave	Slot	Comment
			1	8		
	AD70	DWORD	1	8	1	
	AD74	DWORD	1	8	1	
	AD78	DWORD	1	8	1	
	AD82	DWORD	1	8	1	
	AD86	DWORD	1	8	1	
	AD90	DWORD	1	8	1	
	AD94	DWORD	1	8	1	
	AD98	DWORD	1	8	1	

Profibus

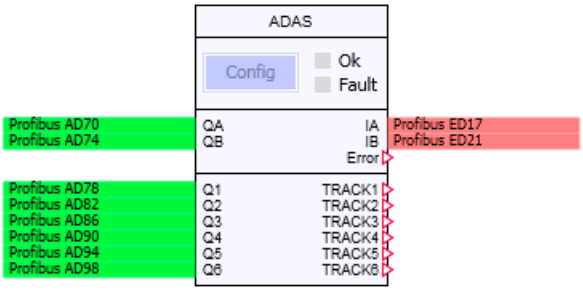
- Profibus
  - System 1
    - [3] L-SC IM-SC/def.DP
    - [6] SLAVE_8109
    - [8] ccADAS
      - [1] Header + 6 Tracks (4 Byte)
    - [32] ET 200S HighFeature (Cu)

Property	Value
Module	Header + 6 Tracks (4 Byte)
Slot	1
Addresses Inputs	EB17 - EB24
Addresses Outputs	AB70 - AB101
Failsafe	No
Pull module	

All I/O data are configured as double words in the coupling and must be connected to the SIMIT component (*QA*, *QB*, *Q1* .. *Qn* or *IA* and *IB*):

Diagram

100% Tahoma 12 F K U



The transferred axis values are available at the analog outputs *TRACKi*, *i* = *2*, ..., *N* of a component of type ADAS. The number *N* of available channels can be specified by scaling the component. A maximum of 28 channels can be specified. You can, for example, connect the *TRACKi* outputs

with the 3D viewer control in SIMIT to animate the 3D view of the machine with current axis values.

Note

The number of transferred axis values is specified by assigning parameters to the *ccADAS* slave. You should set the number in the ADAS component to the same value as defined in the slave to use all channels within the component.

In case of communication failure the integer output *Error* provides information about the cause of the error.

Table 8-36 Error codes for the ADAS component

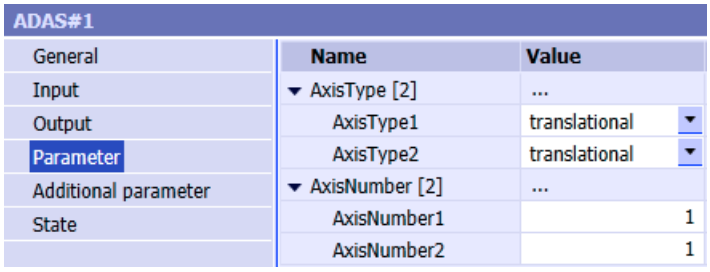
Error	Cause
0	No error
1	No reply to configuration of Track 1
2	No reply to configuration of Track 2
..	
28	No reply to configuration of Track 28
100	No reply to Reset command
101	No reply to Set Communication command

Parameters

The *ADAS* component has the parameter vectors *AxisType* and *AxisNumber*, which have as many elements as the specified number of channels. This allows for each channel to specify which axis of the SINUMERIK-NC is to be transferred and whether the axis moves linearly or is an axis of rotation.

- *AxisType*
  Depending on this parameter assignment, the axis values are provided in mm (translational), angle degrees (rotatory grad) or radians (rotatory rad).
- *AxisNumber*
  This parameter specifies which axis is to be transferred on the corresponding ADAS channel.

The figure below shows the parameters together with their default values:



Note

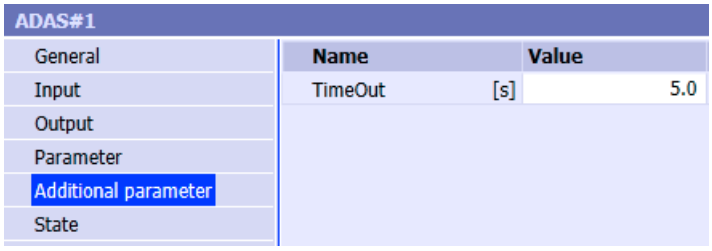
Note that the parameter assignment will be effective only if the *Config* button in the operating window or on the basic symbol is pressed to transfer the configuration to the SINUMERIK while the simulation is running and the SINUMERIK controller is connected.

As an alternative, you may also specify these settings in the SINUMERIK machine data.

Additional parameters

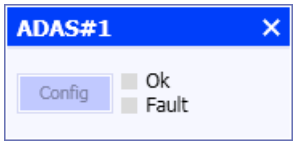
The additional parameter *TimeOut* allows you to specify the period of time after which the component will cancel communication with the SINUMERIK, if the command to transfer the configuration receives no response from the SINUMERIK.

The figure below shows the additional parameter together with its default values:



Operating window

The transfer of the configuration to the SINUMERIK can be triggered in the operating window of the *ADAS* component type by using the *Config* button. While transfer is in progress, the two displays *Ok* and *Fault* are not active. Successful transfer is indicated by the green display *Ok*, and failure is indicated by the red display *Fault*.



The same operating and display elements can be found on the basic symbol and provide the same function.

8.1.6 Controls

8.1.6.1 Controls for displaying signal values

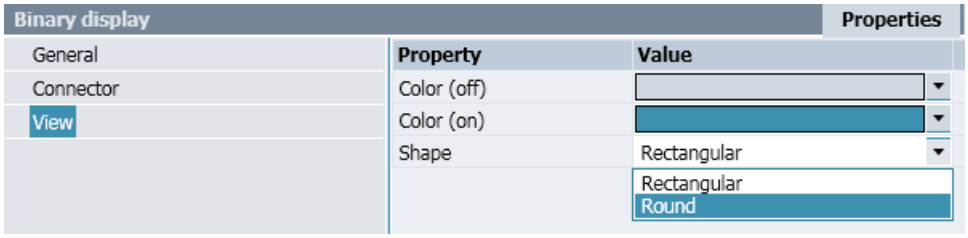
Binary display

Symbol



Function

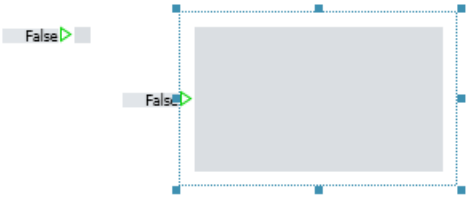
The *Binary display* control is used to display a binary value. The color and shape of the control can be defined in the view properties.



You can select any color for the signal Off state (signal value zero) and On state (signal value one). You can also toggle the shape of the control between *Rectangular* and *Round* in a drop-down box.

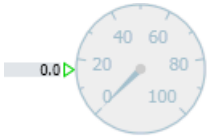


You can change the size of the control as required using the width and height handles on the selection frame. To change the size using the corner handles, hold down the <Shift> key and the size of the control will increase or reduce proportionately.



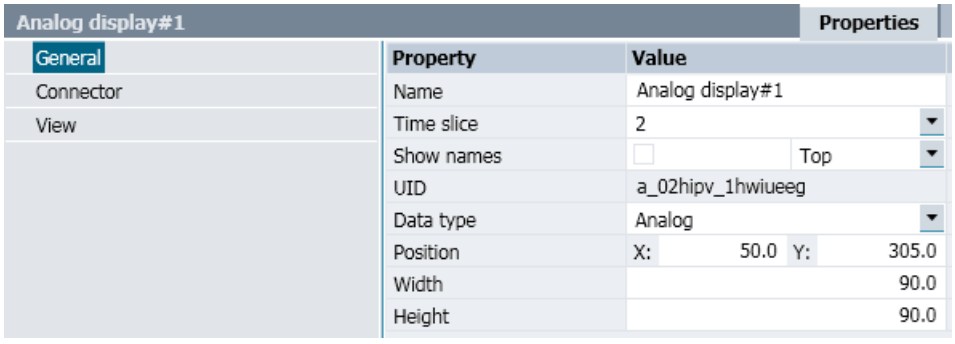
Analog display

Symbol

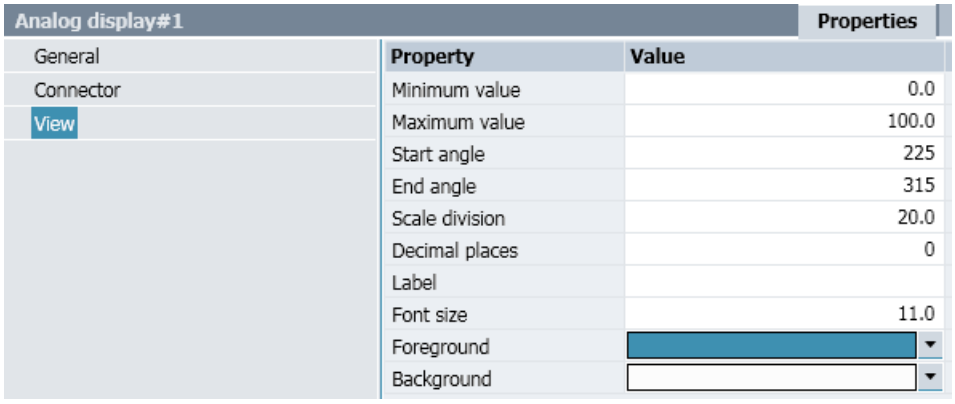


Function

The Analog display control is used to display an analog or integer value in the form of a pointer instrument. The *Data type* of the signal to be displayed can be toggled between *Analog* and *Integer* in the general properties of the control.

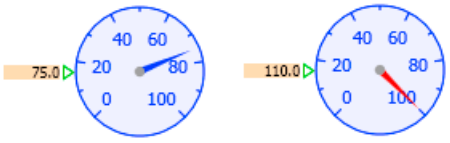


Other settings for the analog display can be adapted individually in the view properties. The figure below shows the property view with the default settings.

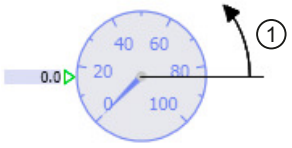


The range of values and the scale, labels and color can all be set as required:

- **Value range**
  The value range is determined by the minimum and maximum values. If the value to be displayed is outside the set range of values, the pointer of the analog display appears in red.



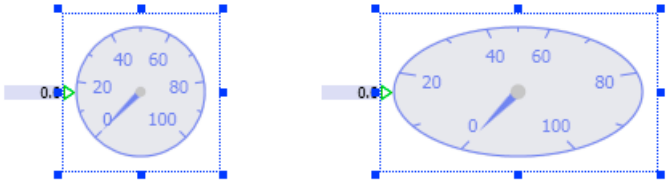
- **Scale**
  Start angle and *end angle* identify the start and end of the scale. The start angle displays the minimum value and the end angle the maximum value. The angle is defined as degrees from the horizontal axis in counterclockwise direction. *Scale division* indicates the subdivision for values on the scale. Scale values are displayed with the specified number of *decimal places*.



① Angle

- **Label**
  You can enter a text in the *Label* property box that will appear in the analog display with the adjustable *Font size*.
- **Color**
  The border, scale and texts are displayed in the adjustable, uniform foreground color. The background color can also be set as the fill color for the control.

You can change the size and thus the round shape of the control as required using the width and height handles on the selection frame. To change the size using the corner handles, hold down the <Shift> key and the size of the control will increase or reduce proportionately.



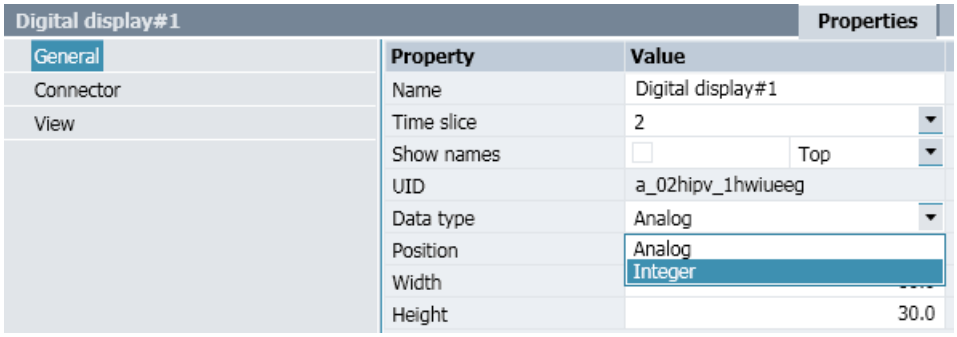
Digital display

Symbol



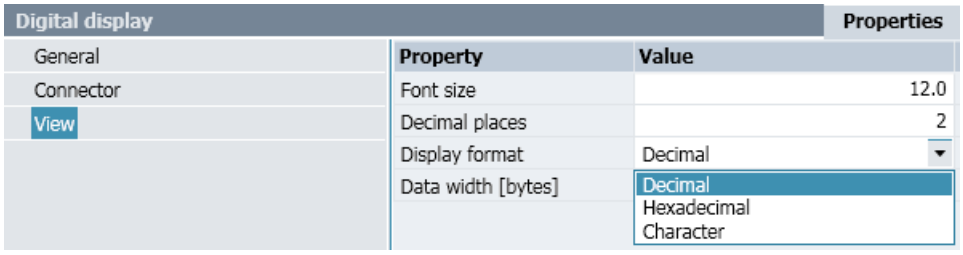
Function

The *Digital display* control is used to display the value of an analog or integer signal. The *Data type* of the signal to be displayed can be toggled between *Analog* and *Integer* in the general properties of the control.



For analog signals you can set the *font size* and the number of *decimal places* to be displayed in the *View* properties dialog.

For integer signals, you can set the *Font size*, the *Display format* and the *Data width* in the property view.



Decimal numbers can be positive or negative. In hexadecimal notation negative numbers are always represented as a two's complement. When converting to hexadecimal numbers you need to specify how many bytes are to be included (1, 2, 4 or 8 bytes). As hexadecimal numbers are displayed with a fixed number of characters, this setting also determines the number of hexadecimal characters displayed (2, 4, 8 or 16 places).

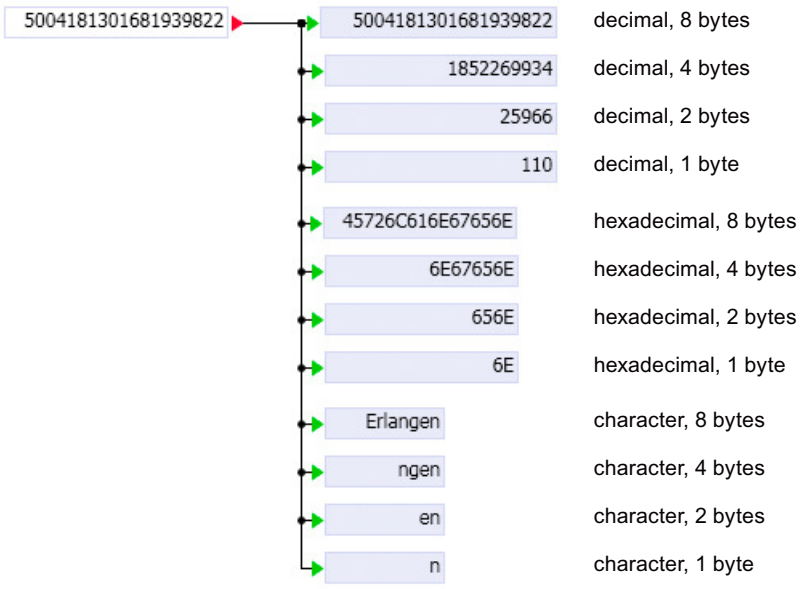
In the *Characters* display format only the specified number of bytes are included, starting with the least significant byte, resulting in a display of 1, 2, 4 or 8 characters. A character corresponding to the coding of the (extended) ASCII code is displayed if it is a displayable character.

Note

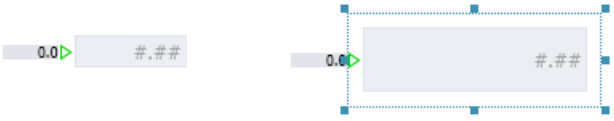
If the data width is set to less than 8, the actual value of an integer input variable might not be displayed in some circumstances.

The data width does not affect the input of a value as the effective value is not limited. However, the way the entered value is displayed will depend on the data width setting.

The figure below shows examples of the effect of different display formats and data widths:

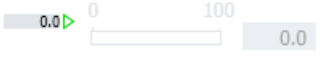


You can change the size of the symbol as required, and thus modify the set font size, using the width and height handles on the selection frame. To change the size using the corner handles, hold down the <Shift> key and the size of the control will increase or reduce proportionately.



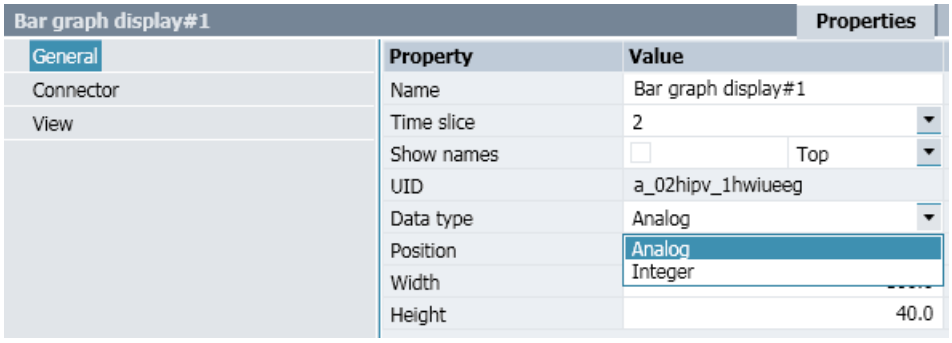
Bar graph display

Symbol

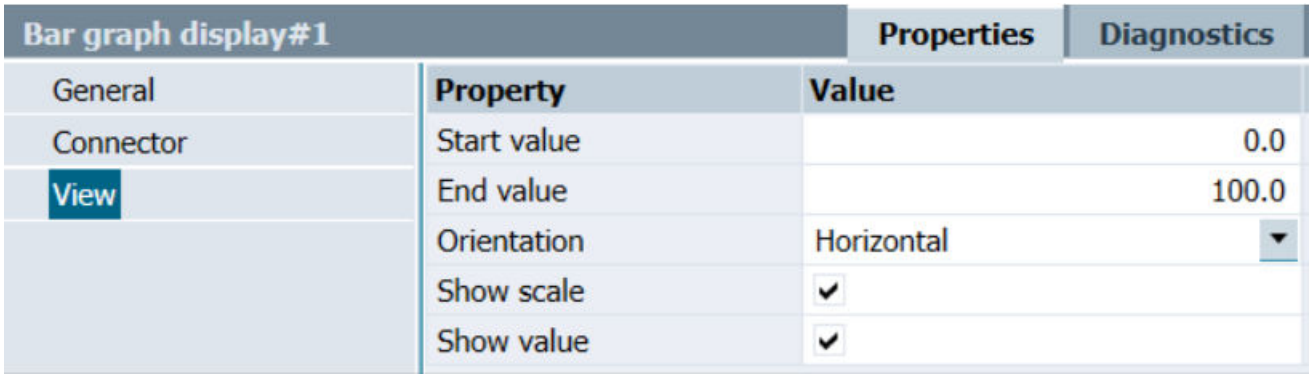


Function

The *Bar graph display* is used to display an analog or integer signal in the form of a bar graph. The *Data type* of the signal to be displayed can be toggled between *Analog* and *Integer* in the general properties of the control.

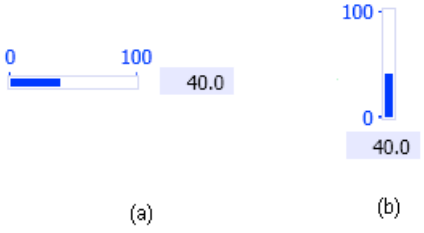


Other settings for the bar graph display can be adapted individually in the property view.



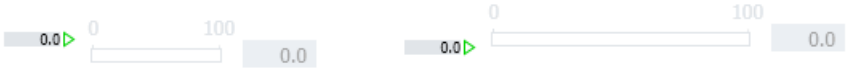
The following properties can be set:

- **Scale**
  The displayed range of values is defined by the *start value* and the *end value*. You can toggle the scale display on and off with the *Show scale* check box.
- **Orientation**
  The possible orientations for the control are *Horizontal* (a) and *Vertical* (b).



- **Bar value display**
  Use the *Show value* check box to toggle the additional numerical bar value display on or off.

You can change the size of the symbol using the handles on the selection frame. This allows you to make the horizontally-aligned bar graph display wider, for example.



8.1.6.2 Controls for entering signal values

Pushbutton

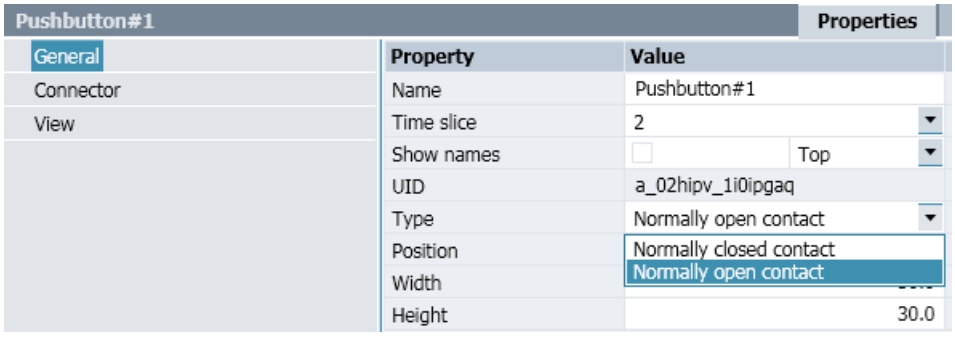
Symbol



Function

The *Pushbutton* control is used to enter a binary signal.

The pushbutton can be defined as *Normally Open* or *Normally Closed* in the general properties. The default setting is *Normally Open*.

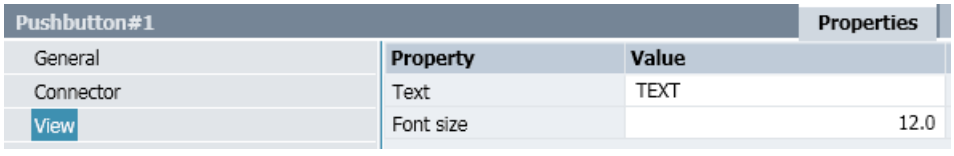


To activate the pushbutton while the simulation is running, simply click the button. The figure below shows the unpressed button (a) and the pressed button (b):


(a)


(b)

As can be seen in the properties window of the figure below, you can specify a *text* for the *pushbutton* that will appear on the button in the configured *font size*.



The handles on the button frame can be used to change the size and thus adjust the size of the specified text, for example. The text is aligned centered in the button (both horizontally and vertically).



Pushbutton with image

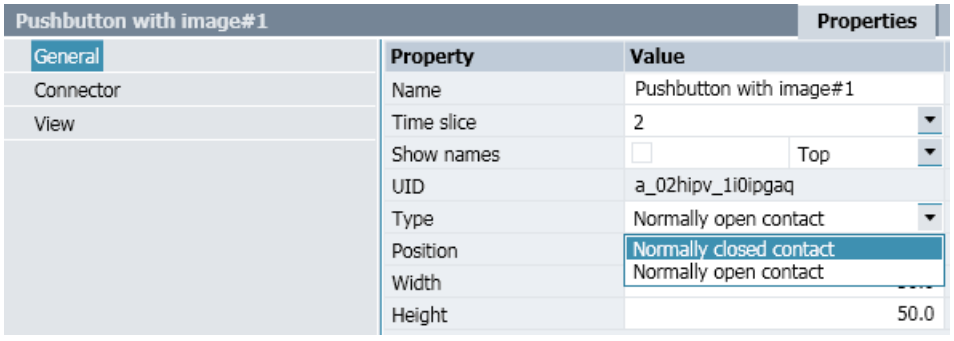
Symbol



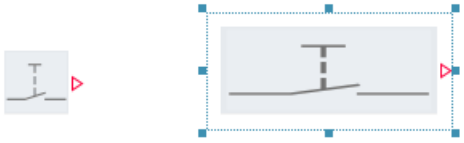
Function

The *Pushbutton with image* is used to enter a binary signal; images are used to represent the button position.

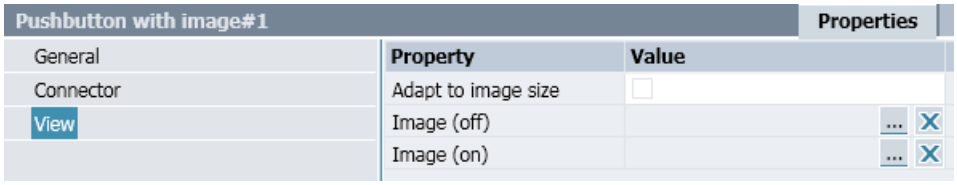
The pushbutton can be defined as *Normally Open* or *Normally Closed* in the general properties. The default setting is *Normally Open*.



To activate the pushbutton while the simulation is running, simply click the symbol. The size of the symbol and thus the size of the sensitive area can be changed using handles on the selection frame.



In the property view you can specify images that represent the *unpressed* (Off) and *pressed* (On) pushbutton while the simulation is running. Click ... to open the dialog for selecting the corresponding graphic files. Click X to delete the selection.



The *Adapt to image size* check box is not selected by default. The selected images will then be adapted to the size of the button, which means the width and height are scaled accordingly. If the check box is selected, the size of the button is matched to the size of the graphic for *Image (not pressed)*.

Switch

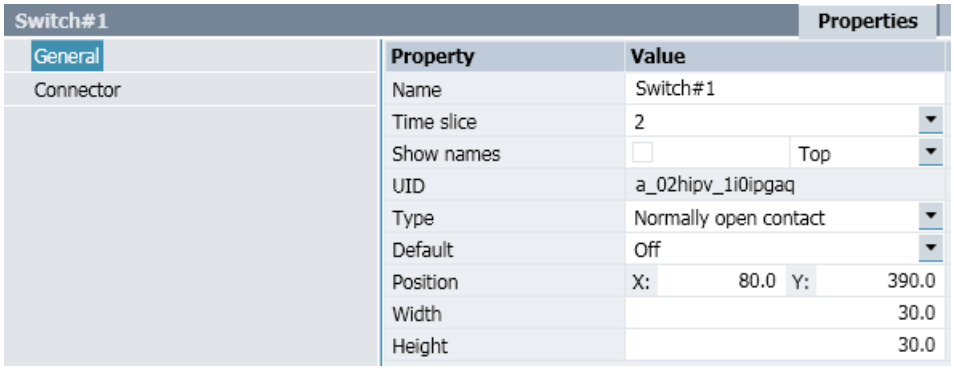
Symbol



Function

The *Switch* is used to switch a binary signal.

The switch can be defined as *Normally Open* or *Normally Closed* in the *Off* or *On* position in the general properties. It is set to *Normally Open* in the *Off* position by default. The selected position takes effect as the *Default* when the simulation starts.



To activate the switch while the simulation is running, simply click the button. The closed switch is represented by a dark-blue border. The table shows the switch as Normally Closed and Normally Open with the default settings *Off* and *On*.

Default setting	NC contact	NO contact
On		
Off		

Handles on the button frame are used to change the size.



Switch with image

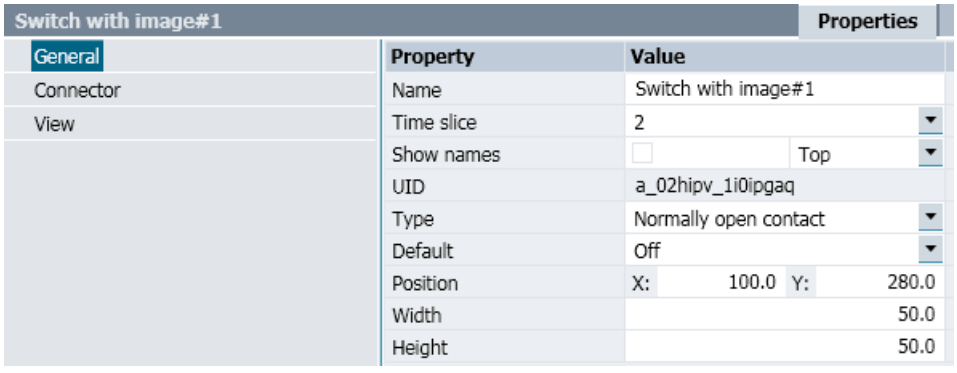
Symbol



Function

The *Switch with image* is used to switch a binary signal; images are used to represent the switch position.

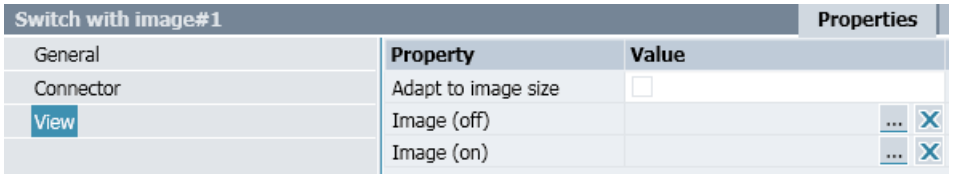
The switch can be defined as *Normally Open* or *Normally Closed* in the *Off* or *On* position in the general properties. It is set to *Normally Open* in the *Off* position by default. The selected position takes effect as the *Default* when the simulation starts.



To activate the switch while the simulation is running, simply click the symbol. The size of the symbol and thus the size of the sensitive area can be changed using handles on the selection frame.



In the property view you can specify images that represent the two possible switch states - *Off* and *On* - while the simulation is running. Click ... to open the dialog for selecting the corresponding graphic files. Click X to delete the selection.



The *Adapt to image size* check box is not selected by default. The selected images will then be adapted to the size of the button, which means the width and height are scaled accordingly. If the check box is selected, the size of the button is matched to the size of the graphic for *Image (off)*.

Step switch

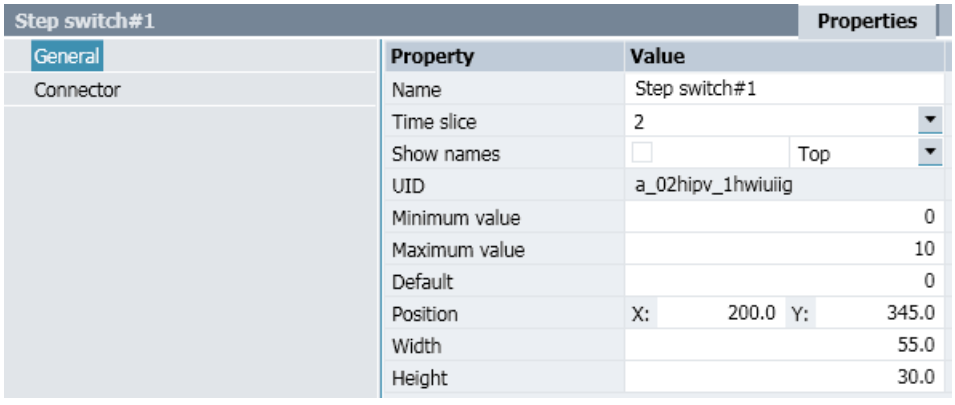
Symbol



Function

The *Step switch* is used to step through the switching of an integer value. The numerical value is increased or reduced by one every time it is switched.

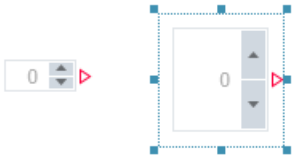
The minimum and maximum values for the switched signal can be defined in the general properties. The default is a ten-step switch with a value range from 0 to10 starting with the switch value 0.



To activate the step switch while the simulation is running, simply click the top ( ) or bottom ( ) button. The switch value that is currently set is displayed in the symbol.



The size of the symbol and thus the size of the two buttons can be changed using handles on the selection frame.



Step switch with image

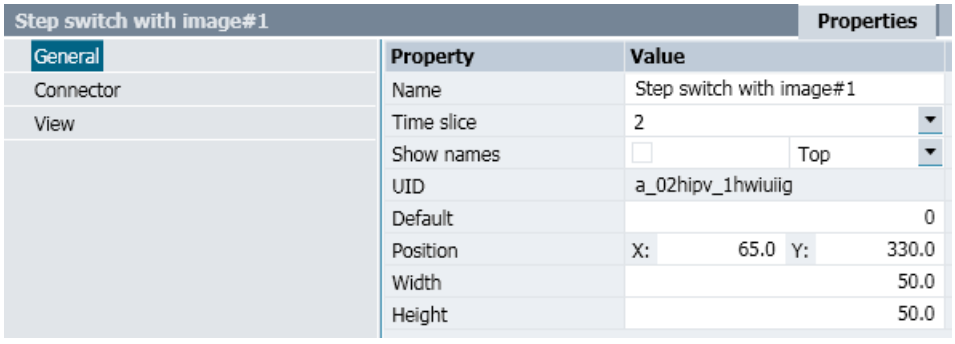
Symbol



Function

The *Step switch with image* is used to step through the switching of an integer value; images are used to represent the switch position.

The switch value from which the step switch starts is set in the general properties. The *default setting* is zero. The numerical value of the switched signal is increased or reduced by one every time it is switched.



The number of switching steps is defined by the number of images. The images are set in the view properties. They change for every step while the simulation is running in the order set in the *Images* list.

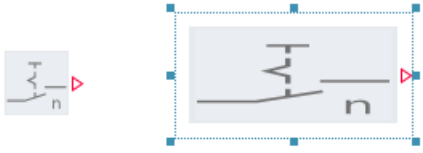
Click ... to open the dialog for selecting the corresponding graphic files. The selected image is added to the list. The order of the images can be changed by swapping an image with the previous ( ) or subsequent ( ) image. Click to delete an image.



To activate the step switch while the simulation is running, simply click the symbol. The *Switchover* is selected via the drop-down list.

In the "Up-Down" parameter setting, the value of the step switch is incremented when you click in the top half of the control and decremented when you click in the bottom half.

Handles on the selection frame are used to change the size of the symbol.



The *Adapt to image size* check box is not selected by default. The images are then adapted to the size of the button, which means the width and height are scaled accordingly. If the check box is selected, the size of the button is matched to the size of the graphic for the first image.

Digital input

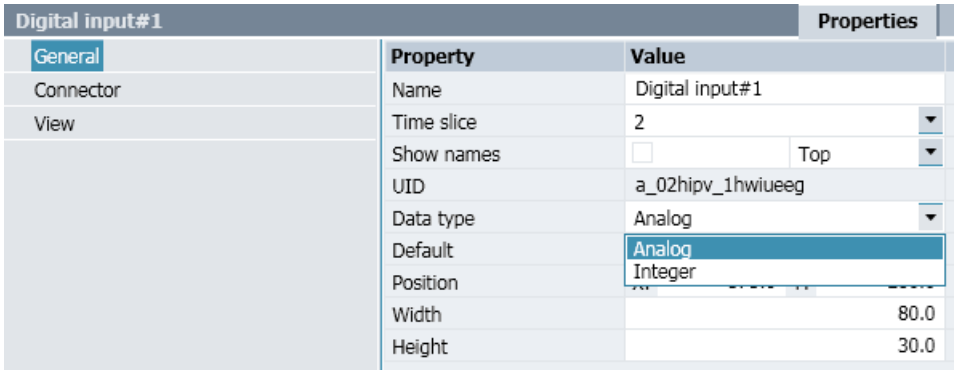
Symbol



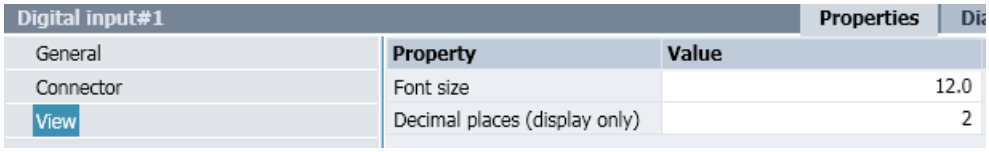
Function

The *Digital input* is used to enter an analog or integer signal value in digital form.

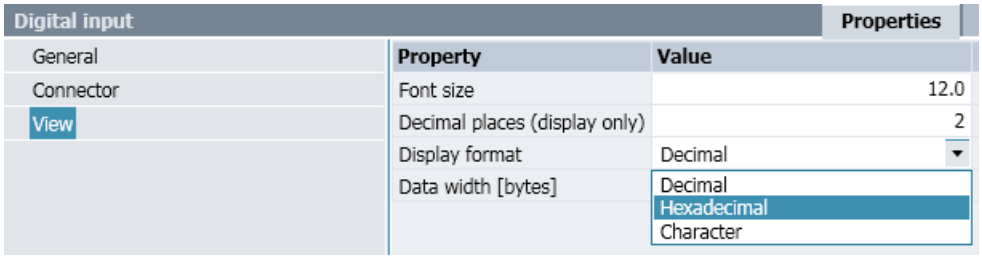
The *data type* of the signal can be set to *analog* or *integer* in the general properties. The *Default* signal value can also be set. It is set to zero by default.



For analog signals you can set the *Font size* and the number of *Decimal places* to be displayed in the View properties.



For integer signals, you can set the *Font size*, the *Display format* and the *Data width* in the property view.



Entries must be made in the specified display format. If the syntax used does not correspond to the specified display format, the input is interpreted as zero.

Decimal numbers can be positive or negative. In hexadecimal notation negative numbers are always represented as a two's complement. When converting to hexadecimal numbers you need to specify how many bytes are to be included (1, 2, 4 or 8 bytes). As hexadecimal numbers are displayed with a fixed number of characters, this setting also determines the number of hexadecimal characters displayed (2, 4, 8 or 16 places).

In the *Characters* display format only the specified number of bytes are included, starting with the least significant byte, resulting in a display of 1, 2, 4 or 8 characters. A character corresponding to the coding of the (extended) ASCII code is displayed if it is a displayable character.

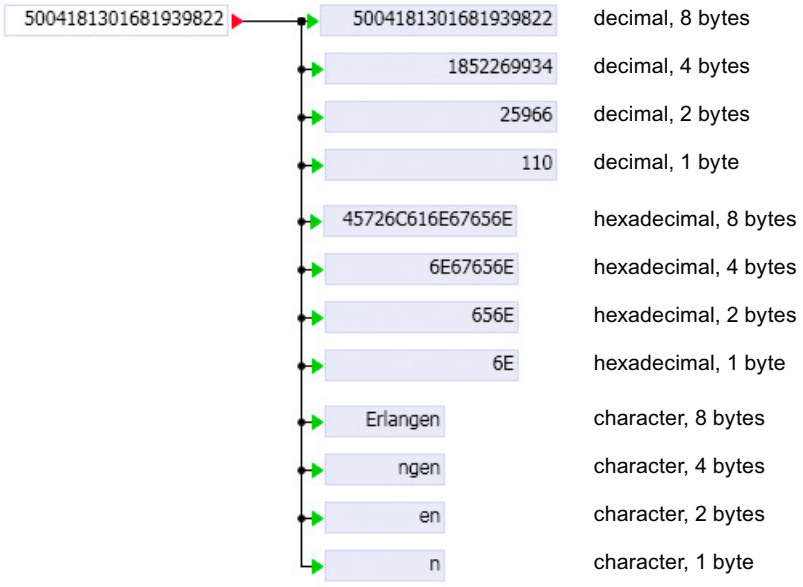
The data width does not affect the input of a value because the effective value is not limited. The specified value is only displayed in accordance with the configured data width.

Note

If the data width is set to less than 8, the actual value of an integer input variable might not be displayed in some circumstances.

When entering values via the Digital Input control, you must use the specified display format. If the entry is syntactically incorrect, it is interpreted as "0".

The figure below shows examples of the effect of different display formats and data widths:



You can change the size of the symbol as required, and thus modify the set font size, using the width and height handles on the selection frame. To change the size using the corner handles, hold down the <Shift> key and the size of the control will increase or reduce proportionately.



The signal values appear right-justified in the symbol while the simulation is running.

Slider

Symbol



Function

The *Slider* is used to set an analog or integer signal.

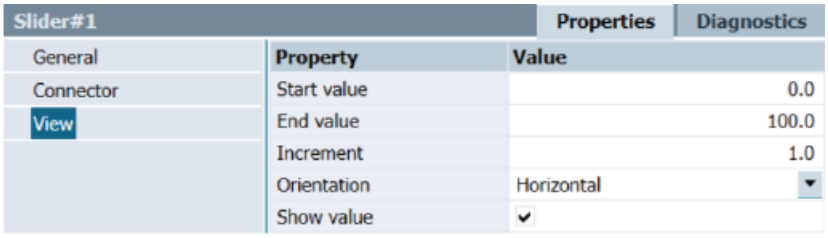
The default setting for the signal value is set in the general properties. The default is zero.



The slider is moved with the mouse while the simulation is running. Position the mouse pointer over the button of the slider, hold down the left mouse button and move the button to the desired position. You can also click to the right or left of the button to increase or reduce the set value by one step.



Other settings for the slider view can be changed individually in the property view.



- **Scale**
  The scale is determined by the *start value*, the *end value* and the *increment*.
- **Orientation**
  The possible orientations for the control are *Horizontal* and *Vertical*.
  The following properties can be set:



- **Bar value display**
  Use the *Show value* check box to toggle the additional numerical display of the signal value on or off.

You can change the size of the symbol using the handles on the selection frame.



8.1.6.3 Miscellaneous controls

Signal splitter

Symbol

Function

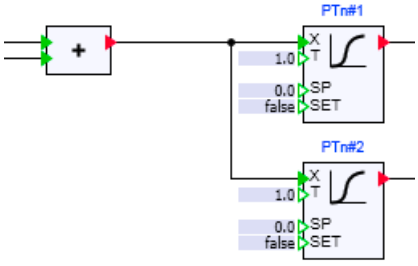
The *Signal splitter* is used to force, which means to set, values at the inputs and outputs of components or coupling signals while the simulation is running using any of the Input controls.

Forcing is already available in the property view for all inputs and outputs of components.

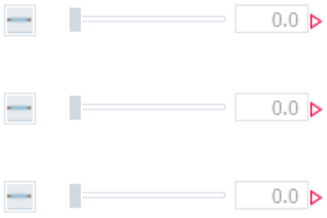
The *Signal splitter* control also allows you to force inputs and outputs using any Input control, as described below by way of example.

Tip: You can use the signal splitter without an assigned Input control and connect it, for example, to the component input or component output that is to be forced. In this case, the signal splitter will perform the same function as the signal splitter in the properties of the input or output.

The figure below shows a section from a chart containing an adder whose output is connected to the inputs of two PTn elements.

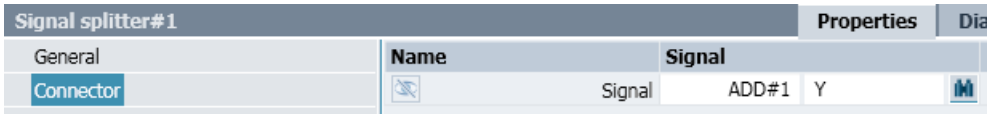


If you want to implement the forcing of the adder output and the two inputs of the PTn elements using sliders, add a slider and a signal splitter to a chart for each input and output to be forced.

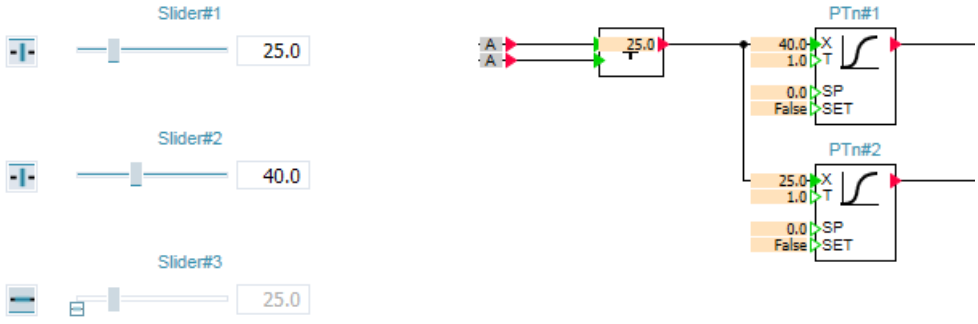


Then open the property view for the first slider, set its connector to invisible and enter the output of the adder as the connected signal. Then open the connector properties for the associated signal splitter. Its connector is always invisible, as can be seen in the figure below.

Here, too, you enter the adder output as the signal.



Connect the other two sliders and signal splitters to the inputs of the two PTn elements in the same way. You can now force the output and the inputs while the simulation is running, as shown by way of example in the figure below.



The signal splitters for the two controls *Slider#1* and *Slider#2* are switched on ( ). Forcing of the adder output and of the input to PTn#1 via the two sliders is thus activated. The signal splitter for *Slider#3* is not switched on ( ), which means the input PTn#3 cannot be forced this way. *Slider#3* is shown as disabled and is also marked with the overlay .

Values for the output and input can now be set using the first two sliders. In the example, the adder output is set to the value 25 and the input of PTn#1 is set to the value 40. The input of PTn#2 is connected to the adder output, and thus takes on its value (25). The (inactive) Slider#3 displays the value of the input connected to it.

The following points should thus be noted when using signal splitters:

- The signal splitter and the associated Input control should be linked to the input or output to be forced.
- Forcing is not activated for signal splitters that are not switched on. The associated input control is shown as disabled and is marked with the overlay . It indicates the value of the connected input or output.
- Switching on the signal splitter activates forcing.

See also

Properties of the inputs (Page 427)

Properties of the outputs (Page 429)

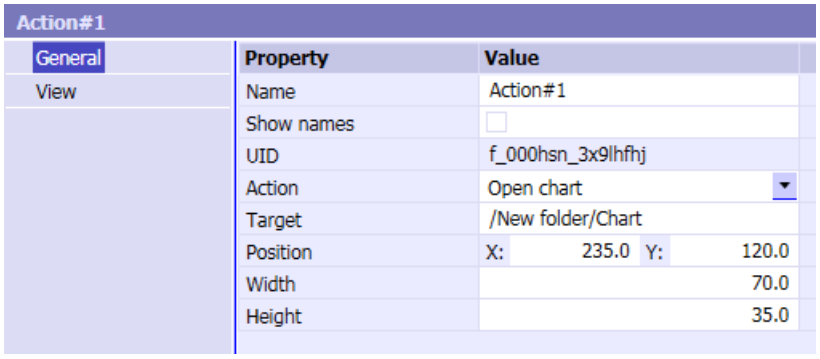
Action

Symbol

Function

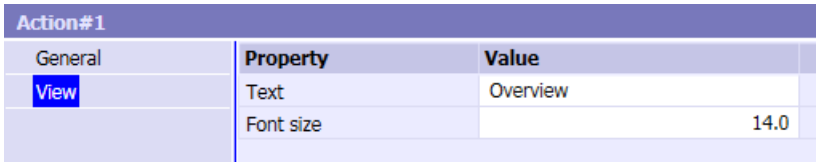
This action opens a chart or a trend.

You can set whether a chart or a trend is to be opened under "Action" In the general properties. As "Target", enter the chart or trend to be opened starting with a slash followed by the complete folder hierarchy separated by slashes. Alternatively, you can drag-and-drop the required chart or trend from the project tree to the "Target" property field.



Click the button to trigger the action while simulation is running.

Text can be specified for the action. This text is shown with the adjustable font size on the button:



The size of the button can be changed and adjusted, for example to adjust it to the size of the specified text. The text is aligned centered in the button (both horizontally and vertically).

8.1.6.4 3D Viewer control

The graphical editor in SIMIT allows you to clearly visualize the behavior of a machine using simple graphical basic elements. Two-dimensional graphics can be used to draw a machine and to show movements of the machine by animating relevant parts of this drawing. The 3D Viewer control gives you the added option of incorporating three-dimensional animated views of a machine in your simulation. The representation of the machine is then clearer, and the movements of the machine are displayed more realistically.

In order to use the 3D Viewer control you must have a three-dimensional geometry model of the machine. This 3D model must be made kinematic, which means it must be designed in such a way that it can be linked to signals from the functional simulation and then execute the animations controlled by these signals in the simulation. The kinematic 3D model includes

elements that are evaluated by the 3D Viewer control and converted into animations of the 3D model.

Like the other controls in the basic library, the 3D Viewer control is inserted in a chart and its parameters are assigned accordingly. It also has connectors with which it can be connected to signals of the functional model. A special feature of the 3D Viewer control is a menu that can be used to adapt the view of 3D model by means of commands.

Data format requirements

In order to use the 3D Viewer control you need a three-dimensional geometry model in VRML V2.0 format. This VRML format can be exported from most CAD systems. In some cases, however, you will also need to restructure the VRML model after export to identify and capture the shapes or shape groups that are to be animated as kinematic simulation points. In terms of the size of a model exported from CAD systems, it is also worth reducing it by eliminating details of the geometry model that are not necessary for visualization.

To restructure the 3D model and make it kinematic, you can use a suitable VRML editor that shows the geometric structure of the 3D model and allows you to modify the VRML code. Information on editing environments for VRML can be found on the web pages of the Web3D Consortium, for example: www.web3d.org.

Animating the 3D model

The 3D Viewer control allows you to animate individual shapes or shape groups of the 3D model in various ways. You can:

- Move and rotate shapes or shape groups in space (translational and rotational movement)
- Reshape shapes or shape groups (size scaling)
- Change the color and transparency of individual shapes

In order to perform these operations, you need to assign appropriate elements or identifiers to the individual shapes or shape groups in the 3D model:

- For movement animations specific motion sensors are added to the 3D model
- For reshaping and changes to the surface of a shape appropriate identifiers are added to the shape definition.

The functioning of the various motion sensors is explained and the other options for modifying models are described in the sections below. Examples are used to show how the various sensors and identifiers can be added to a VRML file.

Animation sensors

A 3D model usually uses sensors to respond to user actions. Sensors in the 3D Viewer control, on the other hand, are routed to connectors. This means you can connect the connectors of the 3D Viewer control to signals from your functional model. This means movements are calculated by the functional model during the simulation and visualized by the 3D Viewer control.

The 3D Viewer control supports the following sensors for animating the 3D model:

- Plane sensors for the translational movement of objects
- Cylinder sensors for rotating objects about the local coordinate axes
- Sphere sensors for rotating objects around a vector

For each motion sensor of the 3D model the following applies:

- At least one higher-level Transform node must exist for a sensor. The sensor can be placed anywhere in the Transform node.
- The position of the sensor alone determines which shapes or shape groups are moved: The Transform node to which the sensor is assigned and all its child nodes are moved by the sensor.
- Routes to sensors are not necessary and are not evaluated.

For each sensor a specific number of connectors is made available in the connector properties of the control. Connect the connectors that perform the desired movement in the 3D model to the corresponding signals in your functional model.

Faulty sensors, which means sensors that are not assigned to a Transform node, are interpreted as not being present. For this reason no animation connectors are available for faulty sensors in the 3D Viewer control properties.

PlaneSensor – the plane sensor

A plane sensor is used to translate an object, which means a shape or a shape group. In the VRML standard plane sensors can be used to move objects in two spatial directions – in the X and Y direction of the local coordinate system. The 3D Viewer control allows a translation in all three spatial directions at each plane sensor. A plane sensor is placed in the VRML model with the keyword **PlaneSensor**.

Once the VRML file has been loaded into the 3D Viewer control, there are three analog connectors available for a plane sensor in the control properties:

Sensorname#TX	for translating the object in the X direction
Sensorname#TY	for translating the object in the Y direction
Sensorname#TZ	for translating the object in the Z direction

If no *Sensorname* is defined for a plane sensor, *TranslationN* is set as the sensor name, whereby *N* is a sequential number for the plane sensor, for example, *N* = 1, 2, ...

The example below defines a cone to which a plane sensor called *ConeSensor* is assigned:

```
#VRML V2.0 utf8
DEF ConeTransform Transform
{
    children
    [
        DEF Cone Shape
        {
            appearance Appearance
            {
                material Material {}
            }
            geometry Cone {}
        }
        DEF ConeSensor PlaneSensor {}
    ]
}
```

Once this VRML file has been loaded into the 3D Viewer control, the following three analog connectors are available in the 3D Viewer control properties for translating the cone:

ConeSensor#TX	(translation in X direction)
ConeSensor#TY	(translation in Y direction)
ConeSensor#TZ	(translation in Z direction)

You can now connect each of these connectors to an analog signal of your functional model to animate the desired movement of the cone. For connectors that have not been connected to signals, no movement of the cone occurs in the corresponding direction.

CylinderSensor – the cylinder sensor

Cylinder sensors are used to rotate an object, which means a shape or shape group by one of the three local coordinate axes. Cylinder sensors should be used as follows to rotate objects about the local coordinate axes X, Y or Z. A cylinder sensor is placed in the VRML model with the keyword **CylinderSensor**. The angle of rotation for an axis is given in degrees.

Once the VRML file has been loaded into the 3D Viewer control, there are three analog connectors available for a cylinder sensor in the control properties:

Sensorname#RX	for rotating the object about the local X axis
Sensorname#RY	for rotating the object about the local Y axis
Sensorname#RZ	for rotating the object about the local Z axis

If no *Sensorname* is defined for a cylinder sensor, *RotationN* is set as the sensor name, whereby *N* is a sequential number for the cylinder sensor, for example, *N* = 1, 2, ...

The example below defines a cone to which a cylinder sensor called *ConeSensor* is assigned:

```
#VRML V2.0 utf8
DEF ConeTransform Transform
{
    children
    [
        DEF Cone Shape
        {
            appearance Appearance
            {
                material Material {}
            }
            geometry Cone {}
        }
        DEF ConeSensor CylinderSensor {}
    ]
}
```

Once this VRML file has been loaded into the 3D Viewer control, the following three analog connectors for rotating the cone are available in the 3D Viewer control properties:

ConeSensor#RX	(rotation about the X axis)
ConeSensor#RY	(rotation about the Y axis)
ConeSensor#RZ	(rotation about the Z axis)

You can now connect each of these connectors to an analog signal of your functional model to animate the desired rotation of the cone. For connectors that have not been connected to signals, no rotation of the cone occurs about the corresponding axis.

SphereSensor – the sphere sensor

Sphere sensors resolve the restriction associated with cylinder sensors of only being able to rotate about coordinate axes. A direction vector is specified for sphere sensors about which the shape is to be rotated. A sphere sensor is placed in the VRML model with the keyword **SphereSensor**. The angle of rotation is given in degrees.

Once the VRML file has been loaded into the 3D Viewer control, there are four analog connectors available for a sphere sensor in the control properties:

Sensorname#R	Angle of rotation
Sensorname#X	X-coordinate of the direction vector
Sensorname#Y	Y-coordinate of the direction vector
Sensorname#Z	Z-coordinate of the direction vector

If no *Sensorname* is defined for a sphere sensor, *SphereSensorN* is set as the sensor name, whereby *N* is a sequential number for the sphere sensor, for example, *N* = 1, 2, ...

The example below defines a cone to which a sphere sensor called *ConeSensor* is assigned:

```
#VRML V2.0 utf8
DEF ConeTransform Transform
{
    children
    [
        DEF Cone Shape
        {
            appearance Appearance
            {
                material Material {}
            }
            geometry Cone {}
        }
        DEF ConeSensor SphereSensor {}
    ]
}
```

Once this VRML file has been loaded into the 3D Viewer control, the following four analog connectors for rotating the cone are available in the 3D Viewer control properties:

ConeSensor#R	Angle of rotation
ConeSensor#X	X-coordinate of the direction vector
ConeSensor#Y	Y-coordinate of the direction vector
ConeSensor#Z	Z-coordinate of the direction vector

You can now connect each of these connectors to an analog signal of your functional model to set the direction vector and animate the desired rotation of the cone.

Scaling objects

If you wish to scale the size of an object, which means a shape or shape group, you need to modify the name of the Transform node to which the object is assigned or the Transform node containing the object to be scaled: Prefix the name of the Transform node with the identifier **SCALE**. Negative scale values flip the object in the scaling axis.

Note

A scale value of zero in two degrees of freedom, which means in two directions, will cause the object to disappear from the animation.

Once the VRML file has been loaded into the 3D Viewer control, there are three analog connectors available for a scaled Transform node in the control properties:

SCALETransformname#SX	for scaling in the X direction
SCALETransformname#SY	for scaling in the Y direction
SCALETransformname#SZ	for scaling in the Z direction

Transformname is the name of the Transform node.

The example below shows a VRML model with a box for which a scale has been defined:

```
#VRML V2.0 utf8
DEF SCALEBoxTransform Transform
{
    children
    [
        Shape
        {
            appearance Appearance
            {
                material Material {}
            }
            geometry Box {}
        }
    ]
}
```

Once this VRML file has been loaded into the 3D Viewer control, the following three analog connectors for scaling the box are available in the 3D Viewer control properties:

SCALEBoxTransform #SX	(scaling in X direction)
SCALEBoxTransform #SY	(scaling in Y direction)
SCALEBoxTransform #SZ	(scaling in Z direction)

You can now connect each of these connectors to an analog signal of your functional model to animate the desired reshaping of the box. For connectors that have not been connected to signals, no scaling of the box occurs in the corresponding axis.

Changing the color and transparency of a shape

In VRML a shape is defined by a Shape node. In order to change the color or transparency properties of a shape, the name of the shape must be modified. Prefix the name of the Shape node with the identifier **RGBT**.

Once the VRML file has been loaded into the 3D Viewer control, there are four analog connectors available for a Shape node prefixed with the identifier RGBT in the control properties:

RGBTShapename#CT	for the shape transparency value
RGBTShapename#CR	for the red component of the shape color
RGBTShapename#CG	for the green component of the shape color
RGBTShapename#CB	for the blue component of the shape color

Shapename is the name of the Shape node.

The shape color is determined by corresponding values for the red, green and blue components. Valid values for a color component are in the range 0, ... ,1. The values for the transparency of the shape are also in the range 0, ... ,1. The transparency value 1 means that the shape is transparent and therefore invisible; the transparency value 0 means that the shape is not transparent.

The example below constructs a cylinder for which an animation of the color and transparency is defined:

```
#VRML V2.0 utf8
Transform
{
    children
    [
        DEF RGBTCylinder Shape
        {
            appearance Appearance
            {
                material Material {}
            }
            geometry Cylinder {}
        }
    ]
}
```

Once this VRML file has been loaded into the 3D Viewer control, the following four analog connectors for the transparency and color of the cylinder are available in the 3D Viewer control properties:

RGBTCylinder #CT	(transparency of the shape)
RGBTCylinder #CR	(red component of the shape)
RGBTCylinder #CG	(green component of the shape)
RGBTCylinder #CB	(blue component of the shape)

You can now connect each of these connectors to an analog signal of your functional model to animate the desired color and transparency of the cylinder.

If you only want to animate the visibility of a shape, simply connect connector *#CT* to a signal. The shape is then displayed in its original color (as defined in the VRML file) and you can switch its visibility off and on by means of the signal values zero and one.

Note

An invisible shape remains invisible even if you change the color values. Only changing the transparency value makes it visible again.

To make an entire group of shapes invisible, move the group to a new transformation node and scale the node in two axis directions to zero or one. You can find additional information on this in section: Scaling objects (Page 607).

If you want to assign the same color to several shapes and animate it, you can utilize the inheritance of properties. The material property of a primary shape can be passed onto any number of other shapes. The primary shape includes the *RGBT* identifier in its name. The resulting four connectors of the primary shape can be used to switch the color (and also the visibility/transparency) of all other shapes that inherit this material property.

This is illustrated by the example below: Two cylinders change color at the same time, but only one has the RGBT identifier.

```
#VRML V2.0 utf8
Transform
{
    children
    [
        DEF RGBTCylinder Shape
        {
            appearance Appearance
            {
                # Definition of the primary material property
                material DEF CylinderColor Material
                {
                    diffuseColor 0.2 0 0.8
                }
            }
            geometry Cylinder {}
        }
    ]
    Translation 2 0 0
}
Transform
{
    children
    [
        Shape
        {
            appearance Appearance
            {
                # Inheritance of the material property
                material USE CylinderColor
            }
            geometry Cylinder {}
        }
    ]
    Translation -2 0 0
}
```

Switching viewpoints

If viewpoints are included in a VRML file, they can be switched both dynamically in the simulation and also by manual operation of the 3D Viewer control. If a VRML file is loaded into the 3D Viewer control, the integer *VIEWPOINT* connector appears in the connector properties of the 3D Viewer control, provided that viewpoints have been defined in the file. The *VIEWPOINT* connector can be set in the value range 1, ..., N, whereby *N* is the number of defined viewpoints.

The example below shows the syntax for two viewpoints called Main View and Front View:

```
DEF MainView Viewpoint
{
    position    -2.076697e3 574.432739 4.039152e3
    orientation    0.192129 -0.980301 4.578408e-2 0.482376
    description    "Main view"
}
DEF FrontView Viewpoint
{
    Position    -78.220909 1.075813e3 4.449046e3
    orientation    -0.631093 -0.427644 0.647179 2.717778e-2
    description    "Front view"
}
```

To switch viewpoints, connect the *VIEWPOINT* connector to an integer signal whose values you can use to switch to the corresponding viewpoint.

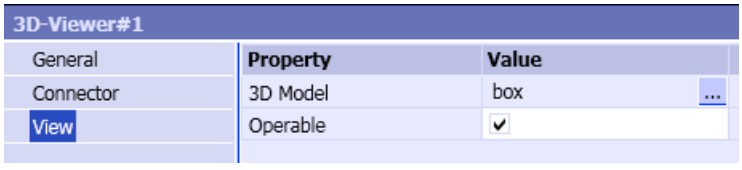
Configuring the 3D Viewer control

In order to display a 3D model in the 3D Viewer control and modify it dynamically, you need to add a 3D Viewer control to the simulation project. Select the VRML file that describes the 3D model in the control view properties. Once you have selected the VRML file, all degrees of freedom of the 3D model are available to you in the control properties as connectors for animating the model. You can then link the connectors to simulation signals from the "Signals" task card, using drag-and-drop for example.

Importing the 3D model

The 3D Viewer control is located in the *Controls* task card in the *Miscellaneous* pane. To add a 3D model to the simulation, you need to place an instance of the 3D Viewer control on a chart. As with all other controls, this is done by dragging and dropping it from the "Controls" task card. The size of the control on the chart – and hence the size of the 3D model – can be changed using the width and height sizing handles on the selection frame.

You can then load the VRML file containing the description of the 3D model in the control view properties.

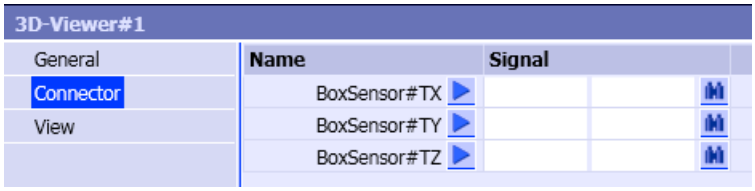


You can use the *Operable* property to operate the 3D viewer control (even when the simulation is stopped) and set the view of the 3D model. You can find more information about this in the section: Simulating with the 3D Viewer control (Page 612).

The scene settings (camera settings) are saved with the chart and are restored when you open the chart.

Linking the connectors to signals

When the VRML file is loaded, all sensors and modifications defined in the file are recorded and made available as connectors in the connector settings.



The connectors that can be used for translating, rotating, scaling, hiding or changing the color of shapes are all provided with a ▶ button. Clicking this button starts an explanatory animation for this connector, making it easy to identify the animated object or its movement axis in the 3D scene. Then link the connectors that execute the animations you require to the corresponding signals of the "Signals" task card using drag-and-drop.

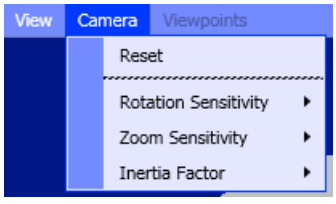
Note

Of the connectors of a sphere sensor, only connector #R, which is used to animate the angle of rotation, has an animation button ▶. When this animation button is activated, a rotation of the object about the X-axis is animated.

Simulating with the 3D Viewer control

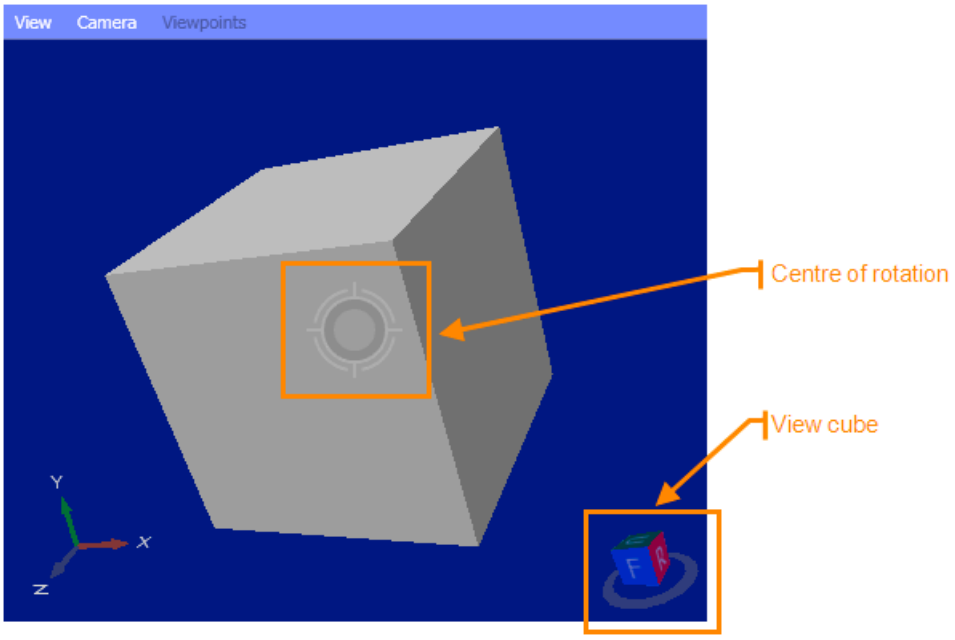
Once the simulation has started you can adjust the view of the 3D model in the 3D Viewer control by means of commands. You can rotate, move or zoom the scene (camera settings) and switch to defined viewpoints. You can make the same adjustments to the scene before the start of simulation by setting the "Operable" view property in the 3D Viewer control.

You can restore the default view at any time with the "Camera > Reset" menu command or the Pos1 key.



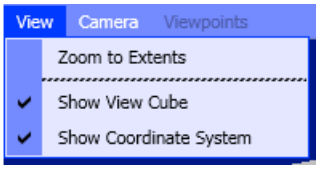
Rotating the scene

You can rotate the scene manually using the mouse or keyboard. Move the mouse pointer over a point on the scene and press the left mouse button. The center of rotation, which is also the viewpoint as seen by the observer (camera), appears in the center of the scene. Hold down the mouse button and rotate the scene in the direction you want. Alternatively you can also rotate the scene vertically or horizontally using the arrow keys.



You can reset the center of rotation/viewpoint by double-clicking an element of the scene.

The view cube in the bottom right corner of the 3D Viewer control shows you the current viewing direction of the 3D scene. Click one side of the cube to reset the scene to the corresponding viewing plane. Double-click to set the opposite viewing plane. You can show and hide the view cube by going to "View > Show view cube".



You can also use the shortcut keys listed in the table below to switch the viewing plane.

Table 8-37 Shortcut keys for switching the viewing plane

Viewing plane	Abbreviation/acronym	Shortcut keys
Front view	F (Front)	Ctrl + F
Back view	B (Back)	Ctrl + B
Left view	L (Left)	Ctrl + L
Right view	R (Right)	Ctrl + R

Viewing plane	Abbreviation/acronym	Shortcut keys
Top view	T (Top)	Ctrl + T
Bottom view	B (Bottom)	Ctrl + B

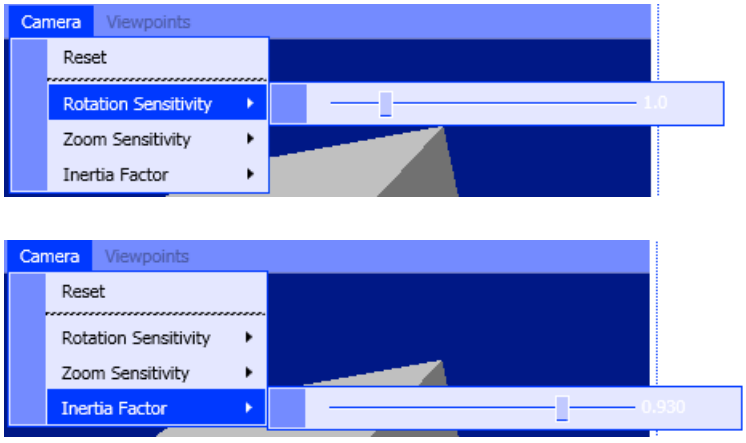
The bottom left corner of the 3D Viewer control shows the coordinate system with the three axes *X*, *Y* and *Z*. You can show and hide this coordinate system by going to "View > Show coordinate system" (see figure above).

Note

The viewing planes are defined in the VRML specification: a Cartesian, left-handed, three-dimensional coordinate system is used in which the positive Y-axis points upwards and the viewer looks from the positive Z-axis towards the negative Z-axis.

When you create the 3D model or export it to a VRML file, make sure that the Y axis is pointing upwards in accordance with the VRML specification.

You can alter the response to rotation with the mouse or arrow keys by going to "Camera > Rotation sensitivity" and "Camera > Inertia". The sensitivity and inertia factor can be reduced or increased by means of a slider. The default setting for both sliders is one.



Swiveling the camera

By panning the camera you can move the scene in the window of the 3D Viewer control, which means move the scene away from or into the center of the window. In contrast to a rotation, if you pan the camera the position of the viewer (camera) remains constant, only the viewpoint (target) changes.

The operating instructions for panning the scene are the same as for rotating, except that you also need to hold down the <Shift> key. You can find information about rotating the scene in the section: Rotating the scene (Page 613).

Zooming the scene

In order to view parts of a scene in more detail you can zoom in or out as required using the Page Up and Page Down keys. The same effect is achieved if you turn the mouse wheel or drag with the left mouse button while holding down the Ctrl key. If you press the <Ctrl> and <Shift> key at the same time and hold down the left mouse button, you can draw an area to be zoomed.

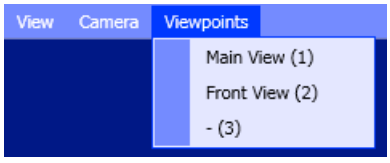
Under "View > Adjust zoom", you can alter the zoom whenever required so that the complete scene fills the screen in the 3D Viewer control. All other camera settings remain unchanged.

You can influence how the scene responds to zooming by selecting Camera > Zoom sensitivity and Camera > Inertia. The sensitivity and inertia factor can be reduced or increased by means of a slider. The default setting for both sliders is one.



Switching the viewpoint

A viewpoint describes a particular viewing position. All viewpoints defined in the VRML file are listed in the 3D Viewer control in the *Viewpoints* menu with their name and a sequential number. If a viewpoint does not have a name, it is listed with the identifier "–". You can switch to a different viewpoint by selecting it in this menu.



8.2 CHEM-BASIC and FLOWNET libraries

8.2.1 Introduction

The FLOWNET and CHEM-BASIC libraries are extensions of SIMIT that provide component types for creating simulations of pipeline networks. You use the component types of CHEM-BASIC, in particular, to create simulations in the chemical and pharmaceutical industries.

By connecting components of these libraries, a model of a pipeline network – a flow network – is created and can be used to simulate the thermodynamic processes in pipeline networks. The FLOWNET and CHEM-BASIC libraries enable use of a special solution method in SIMIT that calculates the flow rates, pressures and specific enthalpies during simulation of pipeline networks.

With CHEM-BASIC the material mixtures are modeled as pseudo ingredients with the corresponding mixture properties (especially heat capacity and density). The individual

components of the flow networks can also be connected to models of the actuator-sensor level (basic library).

Although a model approach based on the physical balance equations underlies the flow networks in SIMIT, the goal is not to enable dynamic process simulations for the design of plant components or plants. Rather, the goal is to provide a physically plausible simulation of the thermodynamic variables in pipeline networks for virtual commissioning. This simulation should be easy to create from components in a graphical interface and should be stable even in extreme situations. Thus, when component types of the FLOWNET and CHEM-BASIC libraries are implemented, a detailed simulation of the physics is not the focus. Rather, the emphasis is on a simple parameter assignment of components and a stable behavior in the flow network.

The component types of the FLOWNET- and CHEM-BASIC libraries can be used to implement flow networks for different media:

- water/steam,
- liquids or
- ideal gases.

With the Modul Component Type Editor (CTE) of SIMIT, you can create your own flow network components and thus specifically expand your flow network library. The flow network solution method can be employed using FLOWNET-specific connection types of the components.

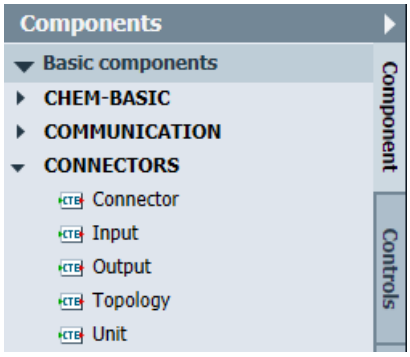
However, existing component types of the CHEM-BASIC library cannot be modified by the user.

Note

A check is made during startup of the simulation to determine whether your SIMIT installation has a license for the FLOWNET- or CHEM-BASIC library. If flow network components are present in your simulation project, that is, components that utilize the solution method for flow networks, the simulation can only be started if you have the license for FLOWNET or CHEM-BASIC.

8.2.2 Topological connector

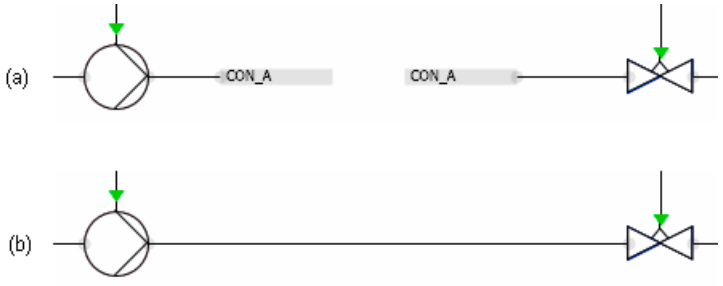
The topological connector *Topology* is available in the *CONNECTORS* directory of the SIMIT basic library; it can be used to create topological connections for flow network components across chart limits.



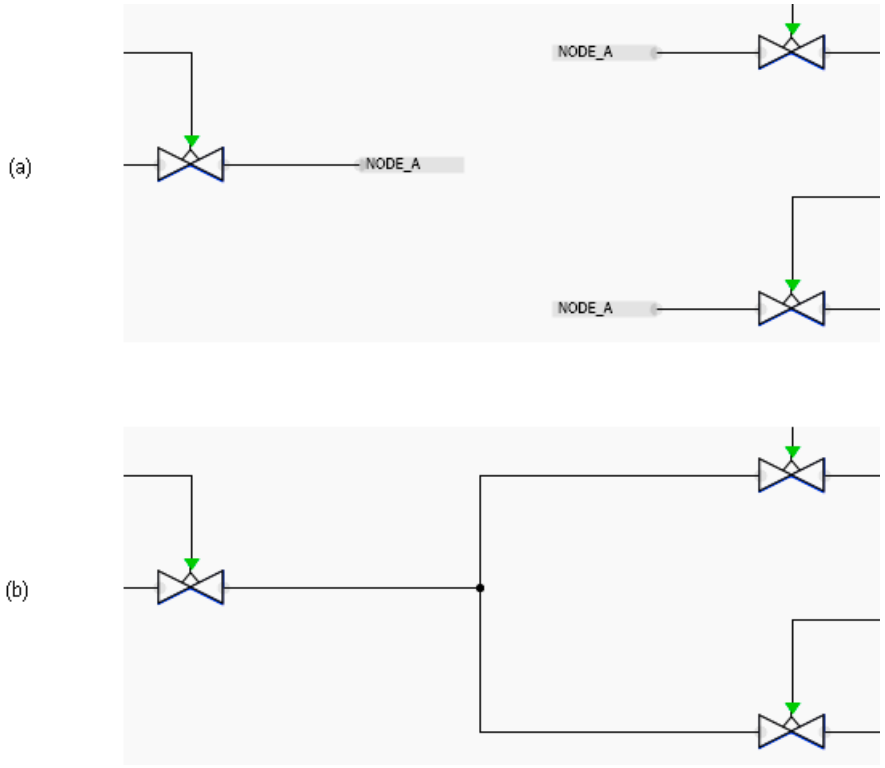
The *Topology* symbol is shown in the figure below:



Use the *Topology* connector to create a topological flownet connection between two or more branch components. Two components are connected by the *CON_A* connector under (a) in the figure below. The connection is functionally identical to the direct connection of both components via a connecting line, as shown under (b) in the figure below:



Three components are connected by the *NODE_A* connector under (a) in the figure below. This configuration is functionally identical to the connection using a connection node that is shown under (b) in the figure below.



8.2.3 Geodetic height

Operating principle

Certain components in the CHEM-BASIC library have an (additional) parameter for geodetic height, *geoHeight*. The geodetic height specifies the height *z* of the equipment. Negative values indicate lower components. The following formula is used to calculate the pressure change resulting from geodetic pressure:

$\Delta P_{geo} = p \bullet g \bullet geoHeight$

A distinction is drawn between the following situations as regards the effect of geodetic height on the simulation:

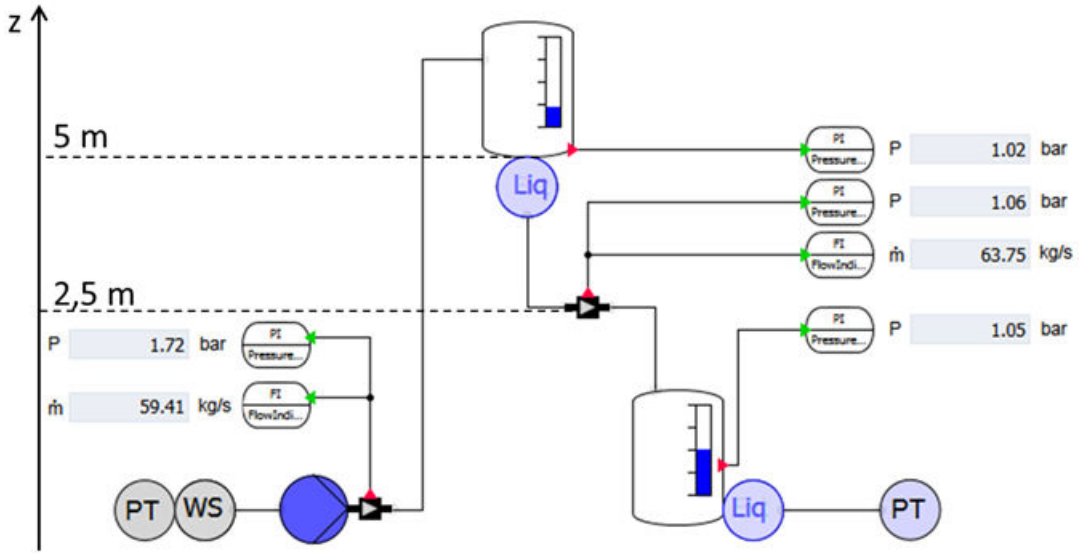
- Components that only connect topologically connected flow.
  Example: *HeatExchangerGeneral*. Connection of two topologically separate flow networks. These flow networks exchange heat, but their pressure and mass flows are separate.
- Components that combine flows from flow networks that are not topologically connected.
  Example: *StorageTankLiquid*. Flows from different flow networks are mixed. The internal pressure in the tank is mapped to the various different flow networks and therefore influences the mass flows.)

Components that only connect topologically connected flows operate with differential pressure only. Geodetic height is not relevant in to this. For these components, the geodetic height is relevant to material properties and internal functions, for example *HeatExchangerGeneral*: Effect on the boiling point of water.

Components that combine flows from topologically separate flow networks work with absolute pressures. The geodetic height therefore affects both the incoming and the outgoing flows. The pressure in these nodes is increased by the geodetic height.

Example

A pump must build up additional differential pressure in order to pump a liquid to a tank that is 5 m higher. From that higher tank, the liquid then flows into a lower tank. A process tag is fitted at a height of 2.5 m. All process tags show the pressure that would actually be measured there.



See also

Centrifuge (Page 698)

Compressor (Page 670)

Condenser (Page 643)

DrumWS – Storage tank for water/steam (Page 753)

ElectricHeatExchanger – Electrically heated heat exchanger (Page 645)

HeatExchangerGeneral (Page 649)

HeatExchangerShellTube – Shell and tube heat exchanger (Page 652)

Mixer (Page 667)

PipeMeasure – Pipe measuring point (Page 662)

Pump (Page 680)

PressureStrainer (Page 726)

Screening device (Page 734)

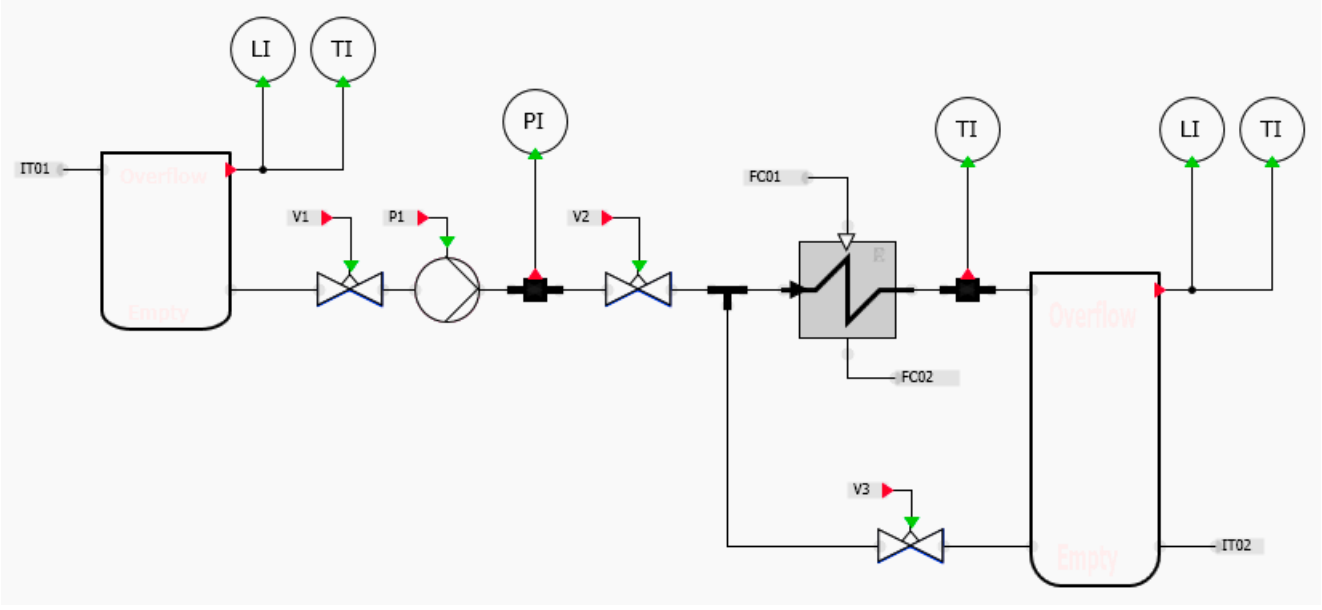
Separator (Page 736)

StorageTankLiquid – Storage tank for liquid media (Page 772)

8.2.4 Flownets

A SIMIT flownet is an interconnection of flownet components used for simulation of thermodynamic processes in pipeline networks. The simulation of flownets is based on a special solution procedure, which is assigned parameters and configured using flownet components. The modeling approach described below limits flownets to homogenous media, but it can be used for liquids, ideal gases or water in either liquid or steam aggregate state.

The FLOWNET library provides component types that can be used to configure or model flownets. As usual in SIMIT, the chart editor is used to model flownets. The symbol used for the flownet component types, such as valves or pumps, is the same as the one normally used in pipeline diagrams. This means you can easily construct a flownet model, as seen in the figure below, in the form of a pipe diagram using the symbols of the component types:



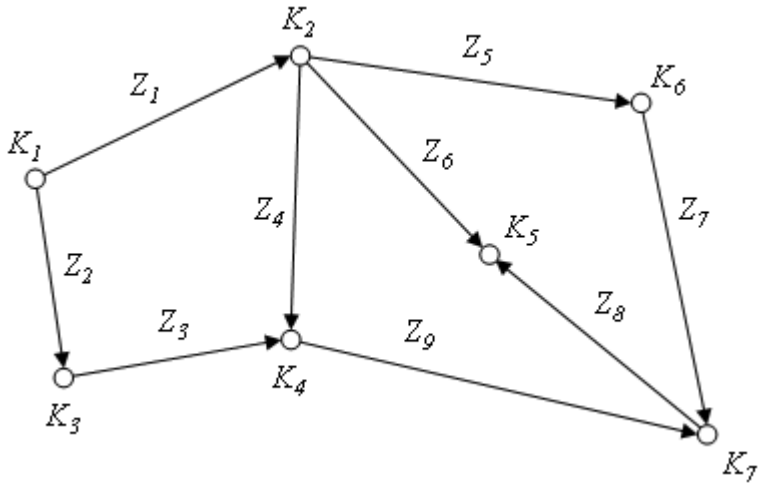
The flownet topology for configuring the flownet solution procedure is derived from the interconnection of flownet components. While the simulation is running, the flownet solution procedure and the flownet components exchange data: calculated values or flownet parameters.

8.2.4.1 Flownet basics

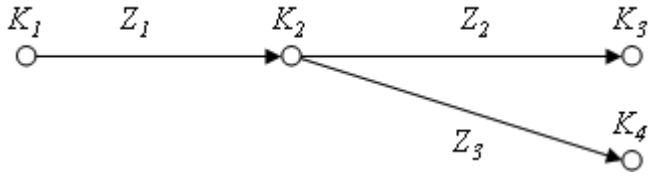
The process for simulating pipeline networks is based on mapping the connection of flownet components to flownets as a graph of nodes and branches. The branches model the flow paths and the nodes model the connections, which means the intersections or joints of the flow paths. The determining variables for the nodes are the pressures and specific enthalpies and the flow for the branches (mass flows).

All physical variables with a vector nature, such as the flow of fluids, are represented in the flownet as one-dimensional variables with direction information, which means as vector quantities (lines of flow). The direction is indicated by the sign. The variables can be numbered arbitrarily for reference.

Flownets can be depicted by graphs, whose branches (edges) form the pipelines with their fittings (valves, pumps, etc.), while their nodes form the pipe joints. The graph is directional, which means the direction of a branch indicates the direction of flow. The graph is also interconnected just like the flownet. The graph represents the topology of the flownet on the level of pipelines and joints. As an example, the following figure shows a graph with 7 nodes K_i and 9 branches Z_i:



A graph, as shown in the following figure, with three branches and four nodes results from the example in the section: Flownets (Page 620).



Nodes K_1, K_3 and K_4 are used to set the boundary conditions for the flownet. For example, the pressure on the connections of both tanks is specified with these external nodes. In contrast, node K_2 is an internal node, for which the relevant variables, such as pressure, are calculated using the flownet solution procedure.

The solution procedure for flownets is based on determining the flow in the branches depending on the pressure, using the momentum balance, and on balancing the flow of matter and enthalpy in the nodes. The state variables for such a system therefore are the mass flows in the branches and the pressures and specific enthalpies in the nodes. Other variables, such as density and temperature in the flownet, can be derived according to the medium in question.

If the flownet components in a branch change neither the rate nor the enthalpy of flow in that branch, the branch is removed from the flownet, which means both nodes are merged to form one node.

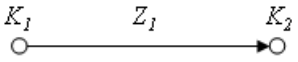
The solution procedure for flownets is a cyclical solution procedure with equidistant time slices that extends the standard solution procedure. Flownet component types can also be used in addition to other component types, such as those in the basic library. Specific connections are used for data exchange between flownet components and the flownet solution procedure. Through these connections, the components receive values calculated by the flownet solution

procedure; these are then used to calculate variables that are sent back to the flownet solution procedure.

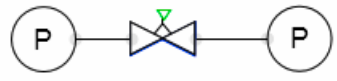
Note

Depending on the structure of the flownet and on the parameter assignment of the flownet components, the calculation of the flownet can become unstable during simulation and the values of the flownet state variables can grow beyond all limits as a result. The stability of flownets is not checked in SIMIT. In this case, you can establish stability by setting a smaller value for the cycle time of the flownet, and/or by modifying the parameters of the flownet balances.

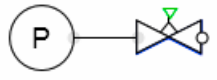
Obviously a flownet must consist of at least one branch with two nodes. The following figure shows the minimal graph.



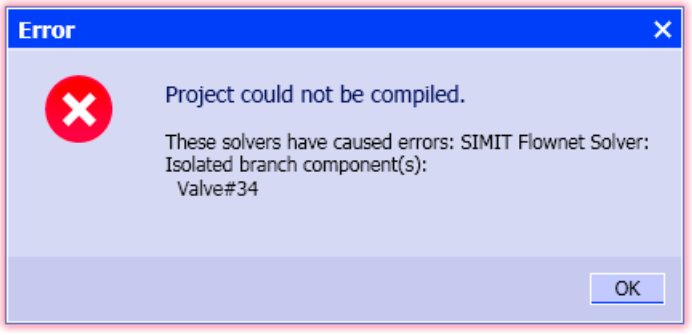
The figure below shows a corresponding minimal flownet. The nodes can be external (as in the figure) or internal.



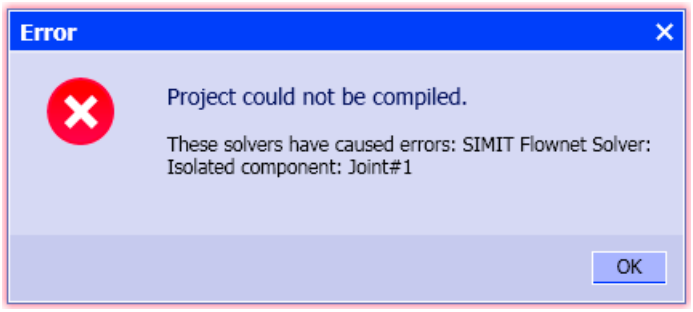
If a branch is not completed with two nodes, as shown in the figure below, an error message is output when you start the simulation and the simulation is not started.



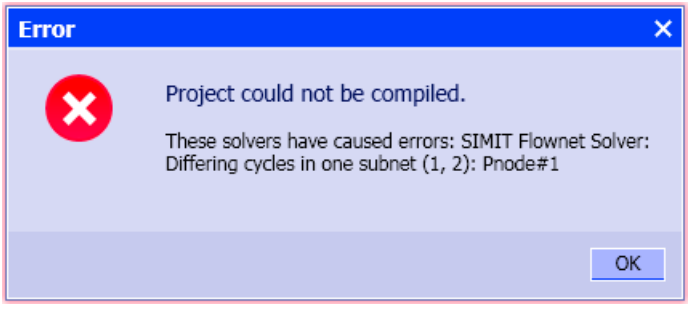
The figure below shows this error message, "*Isolated branch component(s)*", and lists the components in this branch.



A similar error message appears if the flownet only contains a single isolated component. The corresponding graph then consists of only one branch or node and does not meet the minimum requirements for a flownet.



The flownet components and the flownet solution procedure are also processed cyclically in SIMIT. The flownet components must be assigned to a time slice for this purpose. All components of a flownet must be assigned parameters with the same time slice. Otherwise the simulation start is canceled with an error message as the one in the figure below.



8.2.4.2 Variables used in flownets

The mass flows, pressures and specific enthalpies as well as the derived densities and temperatures are considered basic physical variables in flownets. These variables are listed in the following table along with the symbols and units used in this manual.

Table 8-38 Flownet variables

Variable	Symbol	Unit
Mass flow rate	$\dot{m}$	kg/s
Pressure	p	bar (absolute)
Specific enthalpy	h	kJ/kg
Density	ρ	kg/m³
Temperature	T	°C

8.2.4.3 Modeling of flownet branches

It is assumed that branches do not store any mass. The mass flow and density are therefore consistent throughout the entire branch.

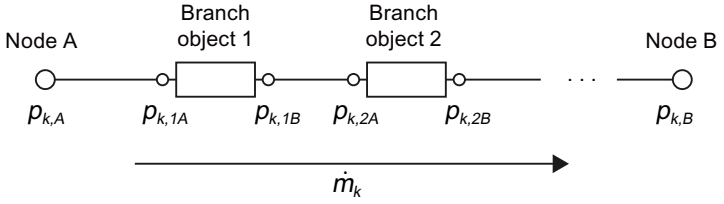
These assumptions are always met for incompressible media. For compressible media no mass is stored in the branch if the branches have "no" volume. The density decreases for compressible media in the direction of falling pressure, which means in the direction of flow; the density change is negligible for slight throttling, so it is acceptable to regard the density as constant.

Thus only the pressures at the connection points need to be considered for branch objects. The relationships between the pressures on either side of a branch object and the mass flow in the branch, such as $\rho \Delta p \sim \dot{m}^2$, are purely of an algebraic nature.

Momentum balance is applied to each branch *k* using pressure forces. Friction forces, acceleration forces and gravity are ignored. With a uniform cross-section A_k in a branch of length L_k, the following applies

$$L_k \frac{d\dot{m}_k}{dt} = A_k \Delta p_k - A_k \sum_{\kappa} \Delta p_{k,\kappa}$$

where $\Delta p_k = p_{k,A} - p_{k,B}$ is the pressure difference over the entire branch *k*, and $\Delta p_{K,k} = p_{K,kA} - p_{K,kB}$ are the pressure differences of the individual branch elements (see the figure below).



If pressure is applied in bars, the rate of change of the mass flow $\dot{m}_k$ is given by

$$\frac{d\dot{m}_k}{dt} = 10^5 \frac{A_k}{L_k} \left(\frac{\Delta p_k}{\text{bar}} - \sum_{\kappa} \frac{\Delta p_{k,\kappa}}{\text{bar}} \right) \frac{\text{kg}}{\text{ms}^2}$$

The length and cross-section of a branch are generally unknown so a reasonable estimate must be made. Assuming for example a cross-section of 0.05 m^2 and a length of 10 m results in a factor that is subsequently described as the momentum factor

$$A_k = 10^5 \frac{A_k}{L_k}$$

of 500 m.

8.2.4.4 Modeling of flownet nodes

The inflow and outflow of the medium are dynamically balanced at each node. Each node is assigned a material balance envelope, which means a volume. The mass inflow and outflow are balanced as well as the enthalpy inflow and outflow as a measure of the energy conversion.

Mass balance for the node

The pressure in a node *i* is determined through the mass balance, which means through the balancing of the inflow and outflow from the branches that are connected with the node:

$$V_i \frac{d\rho_i}{dt} = \sum_{\kappa} \dot{m}_{\kappa}$$

V_i is the volume of the material balance envelope assigned to the node, p_i is the density of the medium within the material balance envelope and the $\dot{m}_\kappa$ are the inflows and outflows. Inflows are positive ($\dot{m}_\kappa > 0$), outflows are negative ($\dot{m}_\kappa < 0$).

Using the equation of state ρ= ρ(*p*,*h*) and assuming an isenthalpic change of state for the pressures in the node, the following applies:

$$\frac{dp_i}{dt} = \left(V_i \left. \frac{d\rho_i}{dp_i} \right|_{h=\text{const}} \right)^{-1} \sum_\kappa \dot{m}_\kappa$$

The term

$$V_i \frac{d\rho_i}{dp_i}$$

is a measure of the compressibility of the medium. Using the compressive modulus K_i

$$K_i = M_i \left(V_i \frac{d\rho_i}{dp_i} \right)^{-1}$$

the following applies for the mass balance:

$$\frac{dp_i}{dt} = \frac{K_i}{M_i} \sum_\kappa \dot{m}_\kappa = c_i \sum_\kappa \dot{m}_\kappa$$

The default setting for the specific compression module $c_i = K_i / M_i$ in the flownet solution procedure is the same for all nodes in the flownet. However, various factors result in increased compressibility c_i of gases and steam in contrast to liquids, and this is taken into consideration for media with higher and lower densities. You can find additional information on this in section: Parameter assignment of flownets (Page 627).

Enthalpy balance for the node

The specific enthalpy is treated as another state variable of a node. In principle, the convective inflow of enthalpy and the existing enthalpy in the node form a mixed enthalpy, which applies to the outflow. The balancing for a node *i* is given by

$$\frac{d(h_i M_i)}{dt} = \sum_\kappa h_\kappa \dot{m}_\kappa$$

where M_i is the mass and h_i is the specific enthalpy of the medium within the material balance envelope. It follows from

$$\frac{dh_i}{dt} = \frac{1}{M_i} \sum_{\kappa \in Z_i} \dot{m}_\kappa (h_\kappa - h_i)$$

that by using the difference in enthalpy only the inflows to the node *i* ($k \in Z_i$) need to be summed.

Just like the mass balance factors c_i, the default thermal factors $m_i = 1 / M_i$ in the flownet solution method are equal for all nodes, but have different values for media with higher and lower densities.

Determining the density of the medium in the nodes

The values for the density in the nodes are calculated from the pressure and specific enthalpy values. The relations used depend on the medium in the flownet.

Water/steam medium

In the case of water/steam, the density in the nodes is calculated using the equation of state for water/steam with pressure p and specific enthalpy h:

$\rho = \rho(p,h)$.

Liquid medium

In the case of liquids, calculation is based on constant density throughout the entire flownet. The default is a density of 997.337 kg/m³.

Calculation

The gas equation $pV = mR_sT$ is used for ideal gas. Using the specific heat capacity c_p the density is given by

$$\rho = \frac{m}{V} = \frac{p \cdot 10^2 \frac{\text{Pa}}{\text{bar}}}{R_S\left(\frac{h - h_0}{c_p} + T_0\right)}$$

with pressure p in bar, the specific heat capacity c_p in kJ/kgK and the specific gas constant R_S in kJ/kgK. If the triple point of water is used for the zero point (T_0, h_0), we can set $T_0 = 273.15$ K and $h_0 = 0$. The density is then calculated using the following relationship

$$\rho = \frac{p \cdot 10^2 \frac{\text{Pa}}{\text{bar}}}{R_S\left(\frac{h}{c_p} + 273{,}15\ \text{K}\right)}$$

In the flownet solution procedure, the specific gas constant has a value of 0.287 kJ/kgK (specific gas constant for dry air).

Determining the temperature of the medium in the nodes

The temperatures used in the FLOWNET library components are also calculated from the values for pressure *p* and specific enthalpy *h*. The equation of state for water/steam medium

$T = T(p,h)$

is used. For ideal gases and liquids the temperature is determined using the specific heat capacity c_p from the specific enthalpy in accordance with

$T = h / c_p$

.

8.2.4.5 Heat exchange with the environment

The heat exchange with the environment is taken into account by a corresponding term in the enthalpy balance equation for the node based on the following formula:

$$\frac{dh_i}{dt} = \frac{1}{M_i}\left(\sum_{\kappa \in Z_i} \dot{m}_\kappa (h_\kappa - h_i) - c_i (T_i - T_{Env}) \right)$$

Where T_{Env} is the ambient temperature, T_i is the temperature of the node and

$c_i = \alpha_i A_i$

is the determining heat transfer factor, which is the product of the heat transfer coefficient α_i and the heat transfer surface A_i.

For calculation of the heat exchange with the environment, use the *JointParam* component as the node.

8.2.4.6 Parameter assignment of flownets

The variables for the branches and nodes of a flow network are calculated with the help of the above-described parameters. These parameters are preset with corresponding values in the flow network solution method but can be changed using special component types of the FLOWNET and CHEM-BASIC libraries.

Component types

You can use the following component types for parameter assignment of networks:

- *NetParam* for general parameter assignment of flownets in FLOWNET (NetParam – network parameter assignment (Page 801)) and in CHEM-BASIC (NetParam – Network parameter assignment (Page 747))
- *NetWS* for parameter assignment of networks with water/steam as the medium in FLOWNET (NetWS – water/steam network parameter assignment (Page 810)),
- *NetLiquid* for parameter assignment of networks with liquids as the medium in FLOWNET (NetLiquid – liquid network parameter assignment (Page 827)) and
- *NetGas* for parameter assignment of networks with ideal gas as the medium in FLOWNET (NetGas – gas network parameter assignment (Page 840)).

You can use the following component types for specific parameter assignment of individual nodes:

- *JointParam* for general parameter assignment of nodes in FLOWNET (JointParam – parameterizable joint (Page 803)) and in CHEM-BASIC (JointParam – configurable pipeline node (Page 637))
- *JointParamWS* for parameter assignment of a node in a network with water/steam as the medium in FLOWNET (JointParamWS – water/steam parameterizable joint (Page 813)),
- *JointParamLiquid* for parameter assignment of a node in a network with liquid as the medium in FLOWNET (JointParamLiquid – liquid parameterizable joint (Page 830)),
- *JointParamGas* for parameter assignment of a node in a network with ideal gas as the medium in FLOWNET (JointParamGas – parameterizable joint (Page 843)),

You can use the following component type for parameter assignment of individual branches:

- *BranchParam* in FLOWNET (BranchParam – branch parameter assignment (Page 802)) and in CHEM-BASIC (BranchParam – Branch parameter assignment (Page 746))

Connection types

The following sections describe the connection types with which you can use parameter assignments in your self-created flow network components in FLOWNET:

- Connector type FLN5 for parameters of a flownet (Page 860)
- Connector type FLN6 for parameter assignment of a branch (Page 861)
- Connector type FLN7 for parameter assignment of an internal node (Page 861)

Flownet media

The following media can be specified for a flownet:

- water/steam,
- ideal gas or
- liquid

Water/steam is the default medium.

Parameters for branches

The momentum factor A can be set for the branches in the flow network. An identical factor A = 450 m is preset for all branches in the flow network solution method. If there are several momentum factors in a branch, the effective factor A in the branch is given by

$$\frac{1}{A} = \sum_{k} \frac{1}{A_k}$$

using the *k* factors A_k.

Node parameters

The following variables can be parameterized for nodes in a flownet:

- Specific compression modulus $c = K / M$
- Thermal factor $m = 1 / M$

Both variables can be individually specified for water and steam as well as liquid and gas.

Transitions between the parameters c_L and m_L and the parameters c_G and m_G for water/steam medium are calculated by the flownet solution method based on the density according to the following scheme:

$$c = \begin{cases} c_L & \text{for} \quad \rho \geq 500\ \text{m}^3/\text{kg} \\ c_G & \text{for} \quad \rho < 500\ \text{m}^3/\text{kg} \end{cases}$$

and

$$m = \begin{cases} m_L & \text{for} \quad \rho \geq 500\ \text{m}^3/\text{kg} \\ m_G & \text{for} \quad \rho < 500\ \text{m}^3/\text{kg} \end{cases}$$

The transition between both parameter values can also be set with a linear transition function:

$$c = \begin{cases} c_L & \text{for} \quad \rho > \rho_L \\ c_G & \text{for} \quad \rho < \rho_G \\ c_L + (c_G - c_L)\dfrac{\rho_G(\rho_L - \rho)}{\rho(\rho_L - \rho_G)} & \text{otherwise} \end{cases}$$

and

$$m = \begin{cases} m_L & \text{for} \quad \rho > \rho_L \\ m_G & \text{for} \quad \rho < \rho_G \\ m_L + \dfrac{m_G - m_L}{v_G - v_L}(v - v_L) & \text{otherwise} \end{cases}$$

with the two reference values $\rho_G = 5$ kg/m³ and $\rho_G = 1000$ kg/m³.

For liquid medium the corresponding variables c_L and m_L are always applicable, and for ideal gas the variables c_G and m_G are applicable.

For heat exchange with the environment, the ambient temperature T_{env} and the heat transfer factor $c = \propto A / M$ can be parameterized. The default temperature is set at $T_{env} = 20$ °C with factor $c = 0$, which means there is no heat exchange with the default settings.

Parameters for liquid medium

If liquid is set as the medium for the flownet, constant density is assumed. The density value to be used can be specified as a parameter.

The specific heat capacity c_p of the medium can be specified as an additional variable for the flownet as well as specifically for each individual node. The default value is 4.18 kJ/kgK.

Parameters for ideal gas medium

For ideal gas as the flownet medium, the gas constant R_S and the specific heat capacity c_p can be specified. The default values are R_S = 0.287 kJ/kgK and c_p = 1 kJ/kgK.

Initialization of variables

Each time the simulation is started, the flownet state variables are initialized as follows:

All branch mass flow rates are initialized with a value of zero ($\dot{m} = 0$). This initialization cannot be changed. The default values for the pressures in the nodes are p = 1 bar, but they can be changed to other values by means of a parameter. The initial value h for the specific enthalpy depends on the medium in the flownet as follows:

- h = 100 kJ/kg for water/steam,
- h = 83.6 kJ/kg for liquid and
- h = 20 kJ/kg for ideal gas.

These initial values can be changed by means of a parameter.

Parameter overview

The table below provides an overview of the available flownet parameters. The signal column lists the names of the input or output signals for the component, the designation column gives the names under which the parameters can be found in the component properties dialog. The default values and units are shown in the last column. If explicit parameters are not set, the defaults for the flownet as described above apply.

Table 8-39 List of flownet parameters

Signal	Title	Description	Default	
			Value	Unit
MEDIUM	Medium	Characteristic number for the medium in the flownet: • water/steam: 0 • liquid: 2 • ideal gas: 1	0	–
CG	sCompressionGas	Specific compression modulus $c = K / M$ for ideal gas or steam	10	bar/kg
CL	sCompressionLiquid	Specific compression modulus $c = K / M$ for liquids	100	bar/kg
MG	FactorThermalGas	Thermal factor $m = 1 / M$ for ideal gas or steam	100	kg^{-1}
ML	FactorThermalLiquid	Thermal factor $m = 1 / M$ for liquids	0.1	kg^{-1}
P_INIT	PressureInit	Initial pressure value	1	bar
H_INIT	sEnthalpyInit	Initial value for specific enthalpy	20 / 83.6 / 100	kJ/kg

Signal	Title	Description	Default	
DENSITY	Density(Liquid)	Density (only applies to liquid)	997.337	kg/m³
T_ENV	TemperatureEnvironment	Ambient temperature T_{env}	20	°C
C_ENV	FactorHeatExchangeEnv	Heat transfer factor $c = \alpha A$	0	kW/K
L_CR	sHeatCapLiquid	Specific heat capacity c_p for liquids	4.18	kJ/kgK
IG_R	GasConstant	Gas constant R_S	0.287	kJ/kgK
IG_CR	sHeatCapGas	Specific heat capacity c_p for gas medium	1.0	kJ/kgK
ST	SmoothTransition	Switch to linear transition of the compression modulus and thermal factor for water/steam	False	
AL	FactorMomentum	Momentum factor A	450	m

8.2.4.7 Coupling simulation model with actuator/sensor level

Coupling simulation model with actuator/sensor level

The physical simulation model created on the basis of the component type libraries CHEM-BASIC and FLOWNET is linked to a control technology by means of an actuator/sensor level.

Signals from the control technology about specific drive components are used in the actuator/sensor level to form the feedback required by the control technology. Drive component types of the SIMIT basic library can, for example, be used for this purpose.

In addition, these drive components calculate, for example, the position for a valve or the standardized speed for a pump.

The physical values of measuring points are read by the actuator/sensor level from the simulation level. These values are then prepared and forwarded in the formats required for the control technology.

This means the actuator/sensor level is connected as follows with the simulation level:

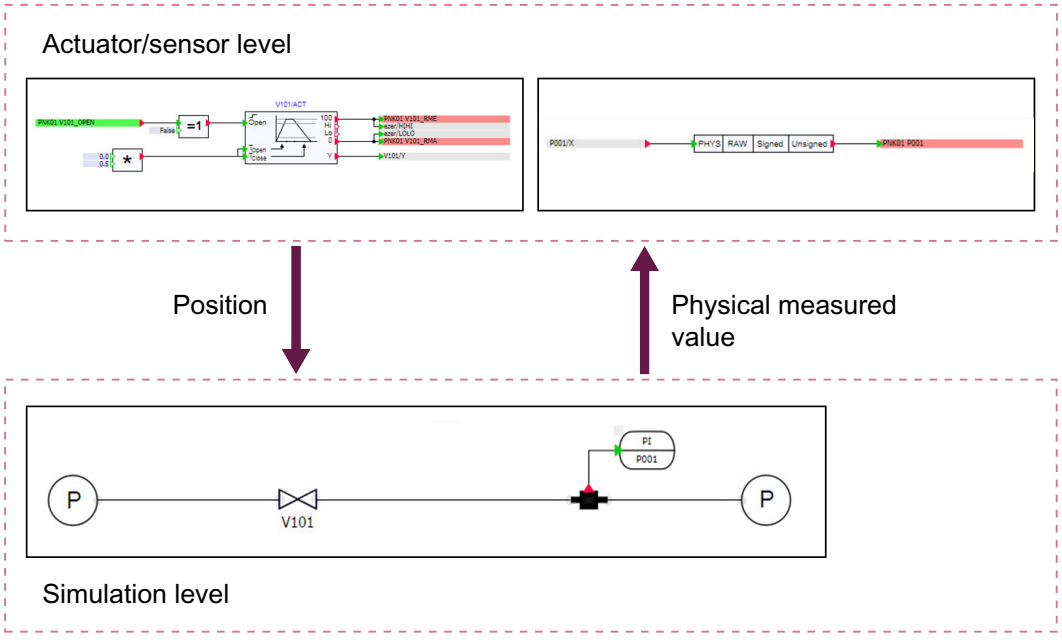


Figure 8-2 Schematic coupling of the actuator/sensor level to the simulation level

The following options are available to link the simulation level to the actuator/sensor level:

- Direct interconnection via signal lines (only possible if actuator-sensor level and simulation level are implemented in a single chart)
- Interconnection of components over signal lines using global connectors
- Implicit interconnection of the actuator/sensor level to the simulation level

Direct connection over signal lines

The direct connection of the actuator/sensor level with the simulation level over signal lines is the simplest type of connection. With this type of connection, all parts of both levels that belong to each other must be located in one chart. This limits the applicability of this type of connection.

A generation of the actuator/sensor level is not practical due to the close linking of both levels in one chart.

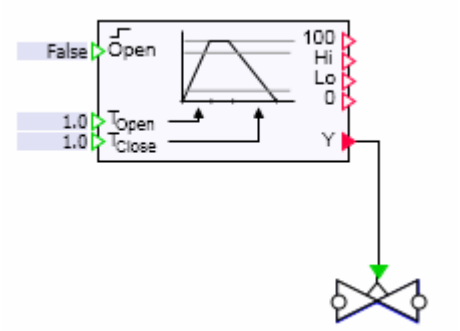


Figure 8-3 Coupling of the valve component type to a drive

Direct connection over signal lines in different charts

A coupling solution that provides a better performance is the connection over signal lines using connectors in different charts. An automated generation of the actuator/sensor level in connection with a manually created simulation level is possible in this instance. The automated generation can, for example, take place by a table import integrated in SIMIT using the SIMIT standard templates. These table imports receive their information from the PCS 7 engineering data.

The major advantage compared to direct connection without connectors is that the actuator/sensor level can be located in a completely separate folder structure where it can be easily swapped or amended.

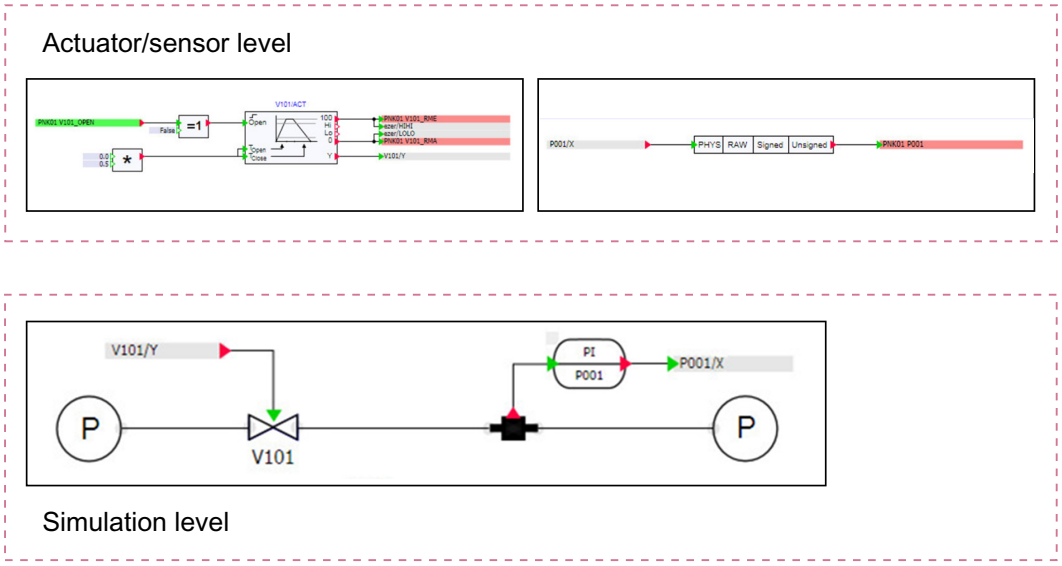


Figure 8-4 Coupling the actuator/sensor level to the simulation level by means of signal lines and connectors

Connection using implicit coupling

When connecting the two levels by means of implicit coupling, the levels can also be located in different charts. It makes sense to also manage these charts in different folders.

The major advantages of this type of coupling compared to coupling with signal lines and connectors are listed below:

- Better overview of charts in the simulation level through simpler display (no connectors necessary)
- The input of the higher level designation (HID) of the components is only necessary in one location (name of the valve, pump or measuring point)
- Parameters set in the model (e.g. valve parameters) are automatically linked to the corresponding HID.

Connection using implicit coupling is achieved by using the same name. To do this, the name and the output of the writing component must be specified at the reading input of a component.

To support the type of coupling for pumps and valves, the input *Position* is given a default. This default is defined as follows:

Name of the component to the connected:	"Valve name"/ACT
Name of the output to be connected:	Y

To use the implicit coupling described above, we recommend assigning the name "Valve name/ACT" to the corresponding drive in the actuator/sensor level. For this purpose, the templates are designed accordingly.

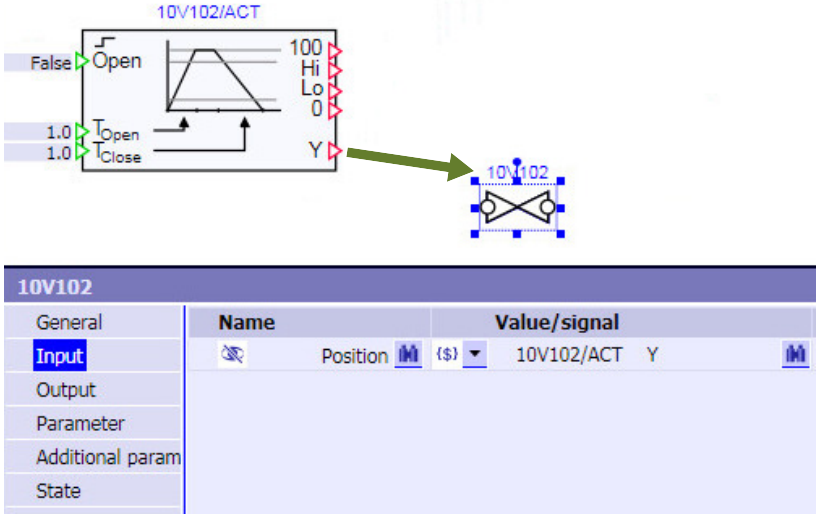


Figure 8-5 Implicit coupling of the valve component type to a drive

In case of an implicit coupling of measuring points, the corresponding reading blocks in the actuator/sensor level must be initialized using templates.

8.2.5 Components of the CHEM-BASIC library

8.2.5.1 Burner

Burner

Symbol

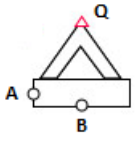


Figure 8-6 Symbol of component type Burner

The component type *Burner* simulates the function of a burner.

A fuel flows through the component.

The component calculates the energy released during combustion and provides it at output Q. This combustion energy can be used to heat the content stored in other components (e.g. *Storage TankLiquid* or *PressureStrainer*).

Function

The *Burner* component is connected to a flow network over the connectors *A* and *B*.

If the fuel is ignited (by means of the input *Ignation* or the "Ignition" button in the operating window), the heat flow released from complete combustion of the fuel is output at the output *Q*. The heat flow is calculated based on the specified heating value and the mass flow. Typical heating values are:

Natural gas	53 MJ/kg
Fuel oil	45 MJ/kg
Gasoline	34 MJ/kg
Raw lignite	10 MJ/kg

There is no actual material conversion of the fuel. The mass flow and the enthalpy of the fuel in the burner are not affected.

As soon as the fuel mass flow falls to zero, the flame extinguishes and there is no longer a heat flow.

The component type *Burner* has different symbols depending on the flame status:

Burner without flame



Burner with flame



Parameters

Parameter name	Description	Unit	Default value
HeatingValue	Heating value of the fuel in MJ/kg	MJ/kg	10.0
Kvs	Flow coefficient k_{VS} with $k_{VS} \geq 10^{-6}$ m^3/h	m^3/h	0.1

Additional parameters

Parameter name	Description	Unit	Default value
Kv0	Valve characteristic value k_{V0} with $k_{V0} \geq 10^{-6}$ m^3/h	m^3/h	0.000001

8.2.5.2 Fittings

ChangeCrossSection

Symbol



Figure 8-7 Symbol of component type ChangeCrossSection

Function

The *ChangeCrossSection* component calculates the pressure drop in a pipe with a change of cross section. The component is configured on the basis of diameter 1 at connection end D1 and diameter 2 at connection end D2.

A constant change in cross section with an angle of 4° is assumed for pressure drop calculations. The component therefore operates either as a nozzle (diameter 1 > diameter 2) or as a diffuser (diameter 1 < diameter 2) depending on the size ratio.

Display

The symbol changes accordingly (direction of flow from D1 to D2).

Nozzle



Diffuser



Parameter

Parameter name	Description	Unit	Default value
Diameter1	Pipe diameter at connection end D1	mm	200
Diameter2	Pipe diameter at connection end D2	mm	300

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure drop	Δp	mbar

Joint – Pipeline node

Symbol



Figure 8-8 Symbol of component type Joint

Function

You can use component type *Joint* to join the three branches connected to its connectors *A*, *B* and *C* in a node. An internal node is added to the flow network with the component *Joint*.

The parameters required for parameter assignment of this node are read from the corresponding *NetParam* block of the flow network. As a result, all nodes of this type located in the same flow network have the same parameter assignment.

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure	p	bar
Specific enthalpy	h	kJ/kg
Density	r	kg/m³
Temperature	T	°C

JointParam – configurable pipeline node

Symbol



Figure 8-9 Symbol of component type JointParam

Function

You can use component type *JointParam* to join the three branches connected to its connectors *A*, *B* and *C* in a node. An internal node is added to the flow network with the component *JointParam*. Unlike with the component type *Joint*, each instance of the component type *JointParam* is configured individually.

Parameters

Parameter name	Description	Unit	Default value
sCompressionGas	Specific compression module for low-density media (gases/ vapors, ρ < 500 kg/m³)	bar/kg	10.0
sCompressionLiquid	Specific compression module for high-density media (liquids, ρ > 500 kg/m³)	bar/kg	100.0
FactorThermalGas	Factor for enthalpy balance calculation of low-density media (gases/ vapors, ρ < 500 kg/m³)	1/kg	100.0
FactorThermalLiquid	Factor for the enthalpy balance calculation of high-density media (liquids, ρ > 500 kg/m³)	1/kg	0.1

Additional parameters

Parameter name	Description	Unit	Default value
PressureInit	Initialization value for the pressure in the node	bar	1.0
sEnthalpyInit	Initialization value for the specific enthalpy in the node	kJ/kg	100.0
TemperatureEnvironment	Ambient temperature	°C	20.0
FactorHeatExchangeEnv	Factor of proportionality for heat exchange of the medium in the node with the environment; no heat exchange occurs when the value zero is set	kW/K	0.0

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure	p	bar
Specific enthalpy	h	kJ/kg
Density	r	kg/m³
Temperature	T	°C

Nozzle

Symbol

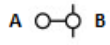


Figure 8-10 Symbol of component type Nozzle

Function

The component type *Nozzle* is used to simulate the pressure drop of a nozzle due to a change in cross section or inlet effects at devices. You can uses this type for connection of pipes to the following component types:

- Tanks
- Heat exchanger
- Centrifuge
- Separator

In principle, you can also connect pipelines to the named components without the *Nozzle* component type. However, it gives you the option of configuring the pressure drop of the connection pipe using the *Kvs* parameter.

The pressure drop calculated in the *Nozzle* component type depends on:

- Size of the nozzle (k_{VS} value)
- Throughflow (volume flow, medium)

The pressure drop across the nozzle is calculated according to the following formula.

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_{VS}^2 \, \rho \frac{\text{kg}}{\text{m}^3}} 12960 \left(\frac{\text{sec}}{\text{h}} \right)^2$$

Whereby:

$\Delta p = p_B - p_A$	The pressure drop across the nozzle in bar
$\dot{m}$	The flow or mass flow in kg/s
ρ	The density of the medium in kg/m³
k_{VS}	The flow coefficient of the nozzle in m³/h

Parameters

Parameter name	Description	Unit	Default value
Kvs	Flow coefficient k_{VS} with $k_{VS} \geq k_{V0} + 10^{-6}$ m³/h	m³/h	10.0

Connection example

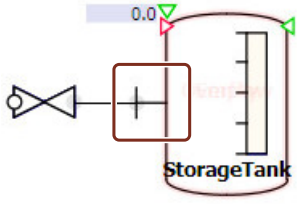


Figure 8-11 Connection example of component type Nozzle with StorageTankLiquid

Orifice

Symbol



Figure 8-12 Symbol of component type Orifice

Function

The *Orifice* component calculates the pressure drop of an orifice in a pipe. The component is configured on the basis of the diameter of the pipe *DiameterPipe* and the diameter of the orifice *DiameterOrifice*.

Parameter

Parameter name	Description	Unit	Default value
DiameterPipe	Diameter of the pipe	mm	200
DiameterOrifice	Diameter of the orifice	mm	40

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure drop	Δp	mbar

Pipe

Symbol



Figure 8-13 Symbol of component type Pipe

Function

The *Pipe* component calculates the pressure drop in a pipe. The pipe is configured on the basis of the pipe length *l*, the pipe diameter *d*, the number of elbows *n*, the ratio of the bend radius to the pipe diameter, and the roughness *k*. Pressure drop largely depends on the Reynolds number *Re*, so you also need to specify the viscosity *η* of the medium.

We differentiate between three main types in the calculation of pipe flow:

- Hydraulically smooth pipes (Re < 65 d/k)
- Transitional range
- Hydraulically rough pipes (Re > 1300 d/k)

Note

The density is only calculated once for each component. Density is therefore calculated as constant throughout the entire section of pipe. If a density change has to be considered, the pipeline must be cut into multiple pieces. These components have to be connected one after the other.

Pipe elbows are always assumed to have a 90° bend. If the ratio of bend radius to pipe diameter RatioRD is greater than 10, elbows are treated like straight pipe. They therefore increase the length of the pipe Length.

Guide values for roughness *k*

Material	State of the pipes	*k* in mm
Asbestos cement		0.05 – 0.1
Cast iron	New	0.25 – 0.5
	Slightly rusted	1.0 – 1.5
	Corroded	1.5 – 5.0
Copper, brass, bronze, aluminum, glass, plastic	Technically smooth	0.001 – 0.0015
Riveted steel		0.5 – 10
Welded steel	New	05 – 0.1
	Rusted	0.4
	Slightly corroded	1.0 – 1.5
	Highly corroded	2.0 – 4.0
Drawn or rolled steel	New	0.02 – 0.1
	Slightly corroded	0.4
	Highly corroded	3.0
Galvanized steel	Smooth	0.07
	Normally galvanized	0.15

Parameter

Parameter name	Description	Unit	Default value
Viscosity	Viscosity of the medium	mPas	0.891
Roughness	Roughness	mm	0.05
Diameter	Diameter of the pipe	mm	200
Length	Length of the pipe	m	1.0
NbrOfElbows	Number of elbows	–	0
RatioRD	Ratio of bend radius to pipe diameter	–	5.0

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure drop	Δp	mbar

The extended operating window also displays:

Variable	Symbol	Unit
Reynolds number	Re	–

8.2.5.3 Graphics

Membrane

Symbol



Function

The Membrane component has no function and is used only for graphic display purposes.

Motor

Symbol



Function

The Motor component has no function and is used only for graphic display purposes.

MotorThreePhase

Symbol



Function

The MotorThreePhase component has no function and is used only for graphic display purposes.

8.2.5.4 Heatexchanger

Condenser

Symbol

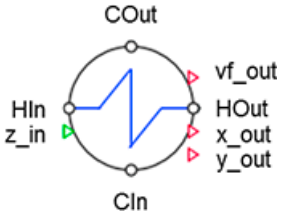


Figure 8-14 Symbol of component type Condenser

Function

The *Condenser* component is a model of a heat exchanger with which the connected flow can be (partially) condensed on the hot side (*HIn*, *HOut*). The heat required for condensing is withdrawn from the flow on the cold side (*CIn*, *COut*). The composition of the hot flow (fraction of the low boiler) is specified over the *z_in* input. The vapor fraction *vf* as well as the composition in the liquid (*x_out*) and in the gas phase (*y_out*) are calculated and made available at the output.

The condenser is parameterized via the heat exchange area *A* and the heat transfer coefficient *k*. The mixture to be condensed is defined over the boiling temperature of the low and high boiler (*TS_LB*, *TS_HB*) as well as over the enthalpies of vaporization (*hv_LB*, *hv_HB*). The size of the boiling lens can be varied using the *S* parameter. This means:

- $S = 0$, maximum size of the boiling lens (see figure)
- $S = T_{HB} - T_{LB}$, no boiling lens (makes no sense physically)

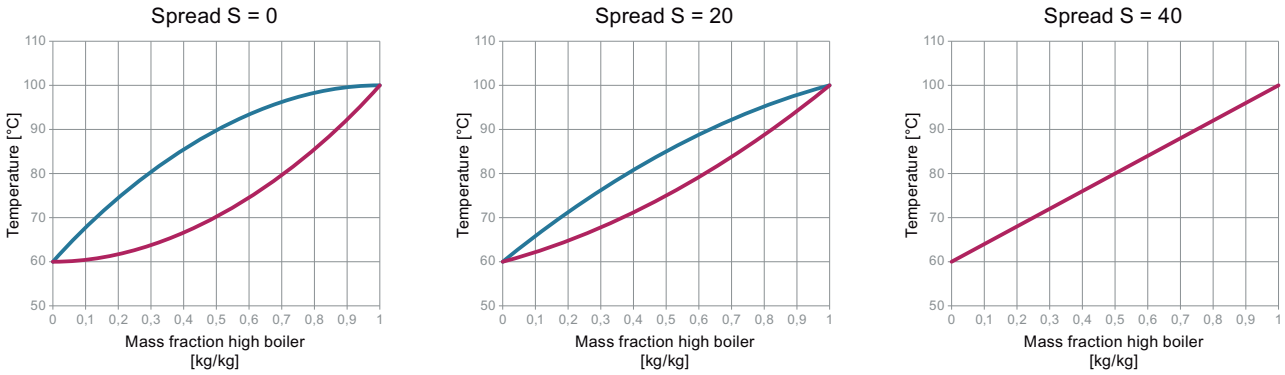


Figure 8-15 Boiling lens as a function of the parameter *S* (spread)

A pressure dependency of the boiling temperatures of the pure substances is neglected.

NOTICE
The flow direction is permanently set on the hot side. A flow is always only possible from the input (*HIn*) to the output (*HOut*). A flow due to a difference in pressure from the output to the input is prevented by a modeled check valve. The flow direction is permanently set on the cold side. A flow is always only possible from the input (*CIn*) to the output (*COut*). A flow due to a difference in pressure from the output to the input is prevented by a modeled check valve. A flownet with the medium *Ideal gas* must be connected at the input on the hot side (*HIn*). A flownet with the medium *Liquid* must be connected at the output on the hot side (*HOut*).

Parameters

Parameter name	Description	Unit	Default value
Area	Heat exchanger area	m^2	10
K	Heat transfer coefficient	$W/(m^2K)$	800
cP_Wand	Heat capacity of the apparatus wall (steel)	kJ/kg	0.5

Additional parameters

Parameter name	Description	Unit	Default value
T_HB	Boiling temperature high boiler	°C	100
T_LB	Boiling temperature low boiler	°C	60
S	Boiling lens spread	–	10
hv_HB	High-boiler enthalpy of evaporation	kJ/kg/K	2300
hc_LB	Low-boiler enthalpy of evaporation	kJ/kg/K	1800
kvs_H	Flow coefficient hot side	m^3/h	10
kvs_C	Flow coefficient cold side	m^3/h	10

Parameter name	Description	Unit	Default value
kv0	Flow coefficient k_V	m^3/h	0.000001
T1	Damping time	s	1
T2	Initialization intervals	–	50
geoHeight	Geodetic height	m	0
zH_Init	Composition on the hot side for initializing	kg/kg	0.8
T_Init	Temperature for initializing	°C	30

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Input temperatures	*T*	°C
Input concentration low boiler	*z*	kg/kg
Output temperatures	*T*	°C
Output concentration low boiler liquid	*x*	kg/kg
Output concentration low boiler vapor	*y*	kg/kg
Vapor fraction in output	*vf*	–

See also

Geodetic height (Page 618)

ElectricHeatExchanger – Electrically heated heat exchanger

Symbol

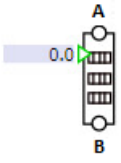


Figure 8-16 Symbol of component type ElectricHeatExchanger

Function

The component type *ElectricHeatExchanger* is used for simulation of an electrical heater in the flow network. The electrical heating power P_{el} in kW is specified with the connector *P_EL*.

It is assumed that the supplied electrical energy P_{el} is completely converted into heat. There are no losses.

$$\frac{dh}{dt} = \frac{1}{\rho V}\dot{Q}$$

The *ElectricHeatExchanger* component is inserted in a flow network using the connectors *A* and *B*. Flows can enter this component type from both ends.

The size of the heat exchanger can be configured with the *Volume* parameter. The result is a delayed temperature change at the output side of the medium when the heating power is switched on or off.

The volume is always considered to be fully filled, which means that mass is neither added nor removed.

The pressure drop across the component is set with the *Kvs* parameter.

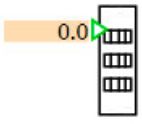
To prevent undesirable states of the simulation, the maximum enthalpy of the outflowing medium can be specified with the *MaxEnthalpy* parameter.

Note

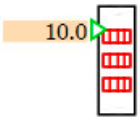
You can also use this component for cooling by specifying a negative heating power P_{el} at the *P_EL* input.

When the *ElectricHeatExchanger* is in heating mode, this is indicated by a change of color in the symbol of the *ElectricHeatExchanger* component.

ElectricHeatExchanger without heating power



ElectricHeatExchanger with heating power



NOTICE

The *ElectricHeatExchanger* calculates only heat transfers and not related density changes. If the *ElectricHeatExchanger* is being operated in a water-steam flow network, these density changes are only calculated in the next flowed-through node.

For this reason, a measuring point in the same branch positioned directly after the *ElectricHeatExchanger* still operates with the unchanged density. The temperature $T = f(p, h)$ calculated within this measuring point is thus not exact.

For an exact calculation, a node is needed between *ElectricHeatExchanger* and the measuring point.

Parameters

Parameter name	Description	Unit	Default value
Volume	Storage volume of the electrical heat exchanger	m^3	0.1
Kvs	Characteristic value for calculation of the pressure drop	m^3/h	360.0
InitialEnthalpy	Initialization enthalpy of the medium	kJ/kg	80.0
MaxEnthalpy	Maximum enthalpy of the medium	kJ/kg	3000.0
geoHeight	Geodetic height; can be changed online	m	0.0

See also

Geodetic height (Page 618)

Evaporator

Symbol

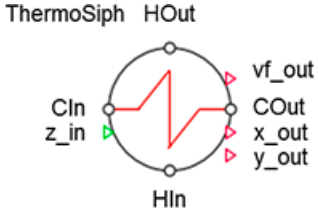


Figure 8-17 Symbol of component type Evaporator

Function

The *Evaporator* component is a model of a heat exchanger with which the connected flow can be (partially) evaporated on the cold side (*CIn*, *COut*). The heat required for evaporation is withdrawn from the flow on the hot side (*HIn*, *HOut*). The composition of the cold flow (fraction of the low boiler) is specified over the *z_in* input. The vapor fraction *vf* as well as the composition in the liquid (*x_out*) and in the gas phase (*y_out*) are calculated and made available at the output.

The evaporator can be used as reboiler of a column with thermosyphon. To do so, the *ThermoSyph* connector must be connected to the corresponding connector of the column. The necessary information (heat flow, vapor fraction, compositions, k_{VS} value of the cold side at the operating point) is exchanged with the column for the thermosyphon via this connector. For the thermosyphon, the current k_{VS} value on the cold side of the evaporator is adjusted so that the mass flow of the thermosyphon is set depending on the differences in density.

The evaporator is parameterized via the heat exchange area and the heat transfer coefficient. The mixture to be evaporated is defined over the boiling temperature of the low and high boiler (*TS_LB*, *TS_HB*) as well as over their enthalpy of evaporation (*hv_LB*, *hv_HB*). The size of the boiling lens can be varied using the *S* parameter. This means:

- $S = 0$, maximum size of the boiling lens (see figure).
- $S = T_{HB} - T_{LB}$, no boiling lens (makes no sense physically)

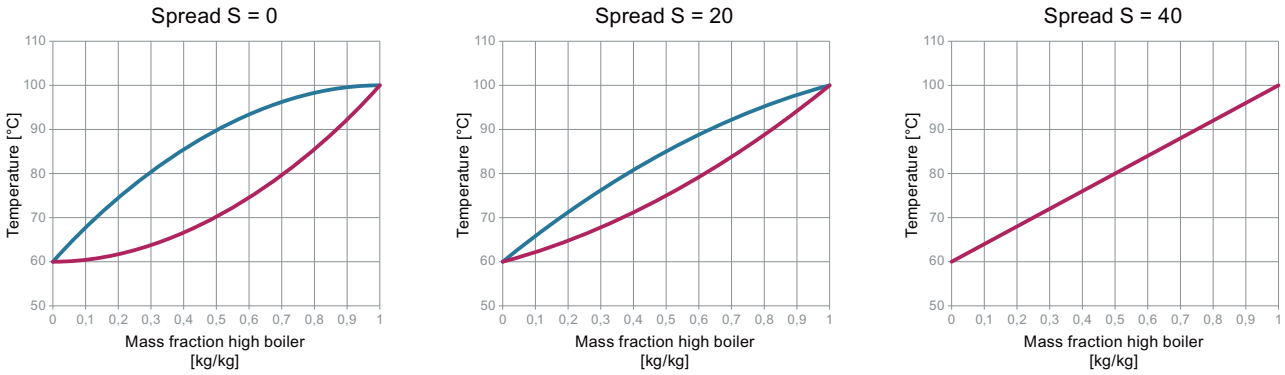


Figure 8-18 Boiling lens as a function of the parameter *S* (spread)

A pressure dependency of the boiling temperatures of the pure substances is neglected.

NOTICE
The flow direction is permanently set on the cold side. A flow is always only possible from the input (*CIn*) to the output (*COut*). A flow due to a difference in pressure from the output to the input is prevented by a modeled check valve. The flow direction is permanently set on the hot side. A flow is always only possible from the input (*HIn*) to the output (*HOut*). A flow due to a difference in pressure from the output to the input is prevented by a modeled check valve. A flownet with the medium *Liquid* must be connected at the input on the cold side (*CIn*). A flownet with the medium *Ideal gas* must be connected at the output on the cold side (*COut*).

Parameter

Parameter name	Description	Unit	Default value
Area	Heat exchanger area	m^2	10
K	Heat transfer coefficient	$W/(m^2K)$	800
cP_Wand	Heat capacity of the apparatus wall (steel)	kJ/kg	0.5

Additional parameters

Parameter name	Description	Unit	Default value
T_HB	Boiling temperature high boiler	°C	100
T_LB	Boiling temperature low boiler	°C	60
S	Boiling lens spread	–	10
hv_HB	High-boiler enthalpy of evaporation	kJ/kg/K	2300

Parameter name	Description	Unit	Default value
hc_LB	Low-boiler enthalpy of evaporation	kJ/kg/K	1800
kvs_H	Flow coefficient hot side	m^3/h	10
kvs_C	Flow coefficient cold side	m^3/h	10
kv0	Flow coefficient k_V	m^3/h	0.000001
T1	Damping time	s	1
T2	Initialization intervals	–	50
geoHeight	Geodetic height	m	0
zH_Init	Composition on the hot side for initializing	kg/kg	0.2
T_Init	Temperature for initializing	°C	60

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Input temperatures	*T*	°C
Input concentration low boiler	*z*	kg/kg
Output temperatures	*T*	°C
Output concentration low boiler liquid	*x*	kg/kg
Output concentration low boiler vapor	*y*	kg/kg
Vapor fraction in output	*vf*	–

HeatExchangerGeneral

Symbol

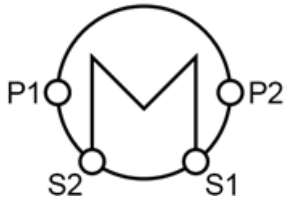


Figure 8-19 Symbol of component type HeatExchangerGeneral

Function

The *HeatExchangerGeneral* component is a model of a general heat exchanger. Heat is exchanged between the product side (PS) and the service side (SS). The heat exchanger is configured on the basis of the heat exchanger area, the heat transfer coefficient and the minimum temperature difference between the product and service side. Heat transferred is calculated using the logarithmic mean temperature difference. If the temperature difference is below the minimum, the flow is adjusted to comply with the minimum temperature difference.

The two mediums are connected at connections P1 and P2 on the process side and connections S1 and S2 at the service side. The heat exchanger operates as either a countercurrent or co-current flow exchanger (see table).

	Process side (PS)	
Service side (SS)	1 → 2	2 → 1
1 → 2	Countercurrent flow	Co-current flow
2 → 1	Co-current flow	Countercurrent flow

NOTICE
The model is not suitable for calculating the phase change at the product or service side when the medium water/steam is connected. Deviations can occur in the calculation of the heat flow and therefore in outlet temperatures. There are therefore three regions for component calculation: • Supercooled liquid – boiling liquid: • Boiling liquid – saturated steam • Saturated steam – superheated steam

Display

During simulation, the direction of heat flow is displayed in color in the symbol of the component. This direction is indicated by the heat flow sign in the heat exchanger calculation.

		
No heat transfer	Heat transfer PS → SS Cooling of product side	Heat transfer SS → PS Heating of product side

If the calculated heat flow is limited by a phase change, the lines will be gray:



If an error occurs in the calculation of the heat flow, the lines will turn red:



This is not a problem if the symbol only appears briefly, for example during a load change. However, if the problem persists over a longer period of time, check the set parameters or the corresponding load points.

Parameter

Parameter name	Description	Unit	Default value
Area	Heat exchanger area	m^2	10
K	Heat transfer coefficient	$W/(m^2K)$	500

Additional parameters

Evaporation/condensation frequently accounts for a large proportion of the heat flow. When you select water/steam as the medium, you can force calculation of condensation/evaporation with the options *PhaseChangePS* (for the process side) and *PhaseChangeSS* (for the service side). This runs a calculation in the wet steam area (region 2). Any heat from superheated steam and supercooled liquid is neglected.

For the calculation to be run, the input temperatures at the heat exchanger must meet the physical requirements. If the option is not selected, the calculation only runs until the wet steam i.e. saturated steam or saturated liquid range is reached. There is no phase change.

Parameter name	Description	Unit	Default value
DTmin	Minimum temperature difference	°C	10
KvsPS	Process side flow coefficient k_{VS}	m^3/h	100
KvsSS	Service side flow coefficient k_{VS}	m^3/h	100
PhaseChangePS	Forced calculation in wet steam region for process side (only relevant when medium is water/steam)	–	False
PhaseChangeSS	Forced calculation in the wet steam region for service side (only relevant when medium is water/steam)	–	False
geoHeight	Geodetic height; can be changed online	m	0.0

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Input temperatures	T_in	°C
Output temperatures	T_out	°C
Heat flow (from service to process side)	Q	kW
Countercurrent mode	Countercurrent	–

See also

Geodetic height (Page 618)

HeatExchangerShellTube – Shell and tube heat exchanger

Symbol

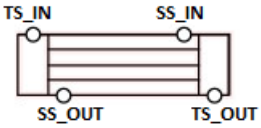


Figure 8-20 Symbol of component type HeatExchangerShellTube

Function

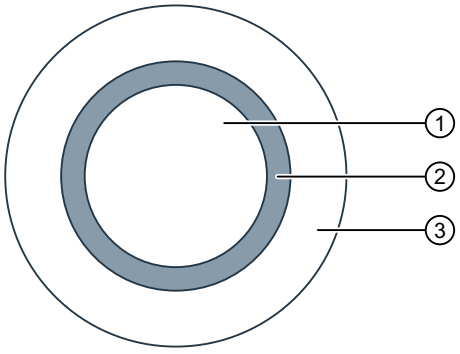
The *HeatExchangerShellTube* component is the model of a single-pass countercurrent-flow shell and tube heat exchanger. The component is therefore configured using corresponding geometric data.

Both media are routed on the tube side via the connectors *TS_IN* and *TS_OUT*, and on the shell side via the connectors *SS_IN* and *SS_OUT*.

If the flow direction is reversed on one side, the heat exchanger works accordingly as a co-current flow heat exchanger.

A total volume for the tube inside and for the shell side is calculated based on the configured geometric data. The transfer area is the sum of the transfer areas of the single tubes.

The figure below shows the resulting single-tube model.



① Tube-side volume
② Tube
③ Shell-side volume

Figure 8-21 Structure of the single-tube model in component type HeatExchangerShellTube

This model is segmented to simplify the calculation. The number of segments is specified with the *NbrOfSegments* parameter. Perfect mixing of the respective medium is assumed within a segment. This approach produces a discretization error.

The number of segments can be selected between 2 and 20.

The heat transfers between the tube-side area, tube material and shell-side area can be set using the *HTC_TS* (tube area → tube) and *HTC_SS* (tube → shell-side area).

Heat losses to the environment via the shell can be parameterized with the parameter *HTC_Shell*. The heat loss is calculated depending on the ambient temperature (*AmbientTemperature* input).

The "stable condition" state in the operating window can no longer be achieved in this case because the heat flow Q (SS-Tube) also includes the heat losses through the shell.

Note

Select the optimal number of segments.

A larger number of segments reduces the risk of a discretization error.

However, too many segments can increase the computational load and cause instabilities.

In the unstable state, the heat exchanger displays the following symbol:



Figure 8-22 Unstable states in HeatExchangerShellTube, autocorrection is active

This is not a problem if the symbol only appears briefly, e.g. during a load change. However, if the problem persists over a longer period of time, check the set parameters or the corresponding load points. You increase stability, for example, by reducing the number of segments or the mass flow.

The type of medium on the shell or tube side results from the connection of the heat exchanger in the flow network and does not have to be configured separately.

The heat capacity of the metal (tubes and shell) is permanently set at 0.46 kJ/(kg·K).

NOTICE

The *HeatExchangerShellTube* calculates only heat transfers and not related density changes. If the *HeatExchangerShellTube* is being operated in a water-steam flow network, these density changes are only calculated in the next flowed-through node.

For this reason, a measuring point in the same branch positioned directly after the *HeatExchangerShellTube* still operates with the unchanged density. The temperature $T = f(p, h)$ calculated within this measuring point is thus not exact.

For an exact calculation, a node is needed between *HeatExchangerShellTube* and the measuring point.

Parameters

Parameter name	Description	Unit	Default value
NbrOfSegments	Number of segments of the heat exchanger	–	6

Additional parameters

Parameter name	Description	Unit	Default value
D_Tube	Outside tube diameter of the single tubes	m	0.016
L_Tube	Length of the tube bundle	m	2.0
N_Tube	Number of tubes	–	150
S_Tube	Wall thickness of the tubes	m	0.0022
Rho_Tube	Density of tubes	kg/m³	7800.0
KvsSS	Flow coefficient on shell side k_{VS}	m³/h	100.0
KvsTS	Flow coefficient on tube side k_{VS}	m³/h	100.0
HTC_TS	Heat transfer coefficient on the tube inside	–	1.0
HTC_SS	Heat transfer coefficient on the tube outside	–	1.0
HTC_Shell	Heat transfer coefficient on the shell side	–	0.0
Shell_Volume_Factor	Factor for calculation of the shell volume depending on the tube volume V_{Shell} = *Shell_Volume_Factor* · V_{Tube}	–	3.0
InitialEnthTS	Initialization enthalpy of the medium in the tube	kJ/kg	120.0
InitialEnthSS	Initialization enthalpy of the medium in the shell	kJ/kg	30.0
geoHeight	Geodetic height; can be changed online	m	0.0

Operating window

The following variables are displayed in the operating window:

Variable	Description	Unit
Massflow SS	Mass flow at the shell side of the heat exchanger	kg/s
DeltaT SS	Temperature change of the medium between entry and exit at the shell side	K
Massflow TS	Mass flow at the tube side of the heat exchanger	kg/s
DeltaT TS	Temperature change of the medium between entry and exit at the tube side	K
Q (Tube->TS)	Heat flow from tube to medium in the tube side	kW
Q (SS->Tube)	Heat flow from medium in shell side to tube and shell	kW

A filled box ("stable condition") indicates that a constant throughflow with stable heat transfer has been reached.

Extended operating window

The currently calculated values (temperatures on the shell side, on the tube side and of the tubes) of each segment are displayed in the extended operating window. In addition, the characteristics of the heat exchanger calculated internally from the configured geometric data are displayed.

The upper part displays the segments with the corresponding temperatures on the tube side (left), shell side (right) and tube material (center). Segments that are not in use are shown as inactive. Six (6) segments are used in the example; you can use a maximum of 20 segments.

The bottom part of the extended operating window shows the characteristics calculated from the configured geometric data.

See also

Geodetic height (Page 618)

HeatPipe – Electrically heated pipe

Symbol



Figure 8-23 Symbol of component type HeatPipe

Function

The component type *HeatPipe* is used for simulation of an electrically heated pipe. The electrical heating power P_{el} in kW is specified with the connector *P_EL*.

It is assumed that the supplied electrical energy P_EL is completely converted into heat. There are no losses.

$$\frac{dh}{dt} = \frac{1}{\rho V}\dot{Q}$$

The *HeatPipe* component is inserted in a flow network using the connectors *A* and *B*. Flows can enter this component type from both ends.

You can configure the size of the heated pipe using the *Volume* parameter. The result is a delayed temperature change at the output side of the medium when the heating power is switched on or off. However, the pipe volume is considered to be perfectly mixed, so that the temperature throughout the entire pipe volume is uniform and is slowly increased or decreased according to the temperature change.

The volume is always considered to be fully filled, which means that mass is neither added nor removed.

The pressure drop across the component is set with the *Kvs* parameter.

To prevent undesirable simulation states, you can specify the maximum possible enthalpy of the outflowing medium using the *MaxEnthalpy* parameter.

Note

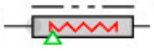
You can also use this component for cooling by specifying a negative heating power P_{el} at the *P_EL* input.

When heating mode is active, this is indicated by a red heating coil in the symbol of the *HeatPipe* component.

HeatPipe without heating power



HeatPipe with heating power



NOTICE
The *HeatPipe* component calculates only heat transfers and not related density changes. If the *HeatPipe* is being operated in a water-steam flow network, these density changes are only calculated in the next flowed-through node. For this reason, a measuring point in the same branch positioned directly after the *HeatPipe* still operates with the unchanged density. The temperature $T = f(p, h)$ calculated within this measuring point is thus not exact. For an exact calculation, a node is needed between *HeatPipe* and the measuring point.

Parameters

Parameter name	Description	Unit	Default value
Volume	Storage volume of the heat pipe	m^3	0.1
Kvs	Flow coefficient for calculation of the pressure drop	m^3/h	360.0
InitialEnthalpy	Initialization enthalpy of the medium	kJ/kg	80.0
MaxEnthalpy	Maximum enthalpy of the medium	kJ/kg	3000.0

Additional parameters

Parameter name	Description	Unit	Default value
geoHeight	Geodetic height; can be changed online	m	0.0

8.2.5.5 Measurements

DifferencePressureIndicator – Differential pressure indicator

Symbol

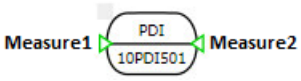


Figure 8-24 Symbol of component type DifferencePressureIndicator

Function

The component type *DifferencePressureIndicator* displays the difference of the pressures specified using its inputs *Measure1* and *Measure2*.

You can specify a high or low limit for the measured value with the *SwitchValue_Plus* or *SwitchValue_Minus* parameter. You can enable limit monitoring with the *Use_Plus* and *Use_Minus* switches. A green binary indicator in the symbol shows that the high or low limit has been reached.

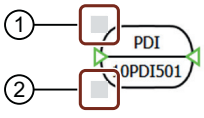
You can set the unit for the analog output and the switching limits with the *UoM* parameter.

If multiple switch values are required, then you must also use multiple measuring points. Alternatively, you can configure additional binary values using logic functions.

For additional linking of measured values, you can make the following outputs of the component visible:

- *AOUT*
  Connection of the analog value with the unit set with parameter *UoM*
- *BOUT_PLUS*
  Connection of the binary value of the high limit
- *BOUT_MINUS*
  Connection of the binary value of the low limit

The state of these switch values is displayed dynamically in the graphic symbol:



① Display of the high limit switch value state
② Display of the low limit switch value state

Figure 8-25 Switch values in the graphic symbol of component type DifferencePressureIndicator

Parameters

Parameter name	Description	Unit	Default value
ICA_Code	MSR code, type of measurement	–	PDI
Use_Plus	Activation of the high limit	–	False
SwitchValue_Plus	Switch value for high limit	see UoM	0.0
Use_Minus	Activation of the low limit	–	False
SwitchValue_Minus	Switch value for low limit	see UoM	0.0
UoM	Returns the measured value in the selected unit. Switch values are also in the selected unit. The following units are available: atm, bar, inHg, inWC, kPa, mbar, mmHg, MPa, mWS, Pa, psi, Torr		bar

Operating window

The following variable is displayed in the operating window:

Variable	Symbol	Unit
Pressure differential of the measuring inputs	Δp	bar mbar kPa Pa psi

DriveIndicator

Symbol



Figure 8-26 Symbol of component type DriveIndicator

Function

The component type *DriveIndicator* only has graphic functions. It is used to illustrate the signal path for the control of drives.

If necessary, you can also use this component directly for controlling drives. To do so, you connect the component to the actuator-sensor level using the *DI* input. The *DO* output is then connected, for example, to the controlling input of the valve or a pump.

Parameter

Parameter name	Description	Unit	Default value
ICA_Code	MSR code	–	YS

FlowIndicator – Flow indicator

Symbol



Figure 8-27 Symbol of component type FlowIndicator

Function

The component type *FlowIndicator* displays the flow specified using its *Measure* input.

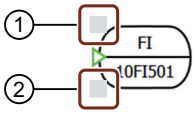
You can specify a high or low limit for the measured value with the *SwitchValue_Plus* or *SwitchValue_Minus* parameter. You can enable limit monitoring with the *Use_Plus* and *Use_Minus* switches. A green binary indicator in the symbol shows that the corresponding switch value has been reached.

You can set the unit for the analog output and the switching limits with the *UoM* parameter.

For additional linking of measured values, you can make the following outputs of the component visible:

- *AOUT*
  Connection of the analog value with the unit set with parameter *UoM*
- *BOUT_Plus*
  Connection of the binary value of the high limit
- *BOUT_Minus*
  Connection of the binary value of the low limit

The state of these switch values is displayed dynamically in the graphic symbol:



① Display of the high limit switch value state
② Display of the low limit switch value state

Figure 8-28 Switch values in the graphic symbol of component type FlowIndicator

Parameters

Parameter name	Description	Unit	Default value
ICA_Code	MSR code, type of measurement	–	FI
Use_Plus	Activation of the high limit	see UoM	False
SwitchValue_Plus	Switch value for high limit	–	0.0
Use_Minus	Activation of the low limit	see UoM	False
SwitchValue_Minus	Switch value for low limit	–	0.0
UoM	Returns the measured value in the selected unit. Switch values are also in the selected unit. The following units are available: g, kg, l, lb, m^3, ml, t per s, min, h, d	–	kg/s

Operating window

The following variable is displayed in the operating window:

Variable	Symbol	Unit
Mass flow	$\dot{m}$	kg/s kg/h t/h
Volumetric flow	$\dot{V}$	m^3/h l/h

LevelIndicator – Level indicator

Symbol



Figure 8-29 Symbol of component type LevelIndicator

Function

The component type *LevelIndicator* displays the level specified using its *Measure* input.

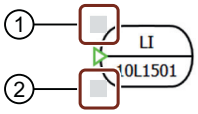
You can specify a high or low limit for the measured value with the *SwitchValue_Plus* or *SwitchValue_Minus* parameter. You can enable limit monitoring with the *Use_Plus* and *Use_Minus* switches. A green binary indicator in the symbol shows that the high or low switch value has been reached.

You can set the unit for the analog output and the switching limits with the *UoM* parameter.

For additional linking of measured values, you can make the following outputs of the component visible:

- *AOUT*
  Connection of the analog value
- *BOUT_Plus*
  Connection of the binary value of the high limit
- *BOUTMinus*
  Connection of the binary value of the low limit

The state of these switch values is displayed dynamically in the graphic symbol:



① Display of the high limit switch value state
② Display of the low limit switch value state

Figure 8-30 Switch values in the graphic symbol of component type LevelIndicator

Parameters

Parameter name	Description	Unit	Default value
ICA_Code	MSR code, type of measurement	–	LI
Use_Plus	Activation of the high limit	see UoM	False
SwitchValue_Plus	Switch value for high limit	–	0.0
Use_Minus	Activation of the low limit	see UoM	False
SwitchValue_Minus	Switch value for low limit	–	0.0
UoM	Returns the measured value in the selected unit. Switch values are also in the selected unit. The following units are available: ft, in, m, mm, %		m

Operating window

The following variable is displayed in the operating window:

Variable	Symbol	Unit
Level	l	m %

NozzleMeasurement

Symbol



Figure 8-31 Symbol of component type NozzleMeasurement

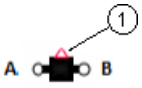
Function

The *NozzleMeasurement* component is used for connecting measuring points (e.g. *LevelIndicator* or *TemperatureIndicator*) to another component (e.g. *StorageTankLiquid* or *PressureStrainer*).

The component transmits the values pending at input *A* to the output *B*. The values remain unchanged

PipeMeasure – Pipe measuring point

Symbol



① Measure

Figure 8-32 Symbol of component type PipeMeasure

Function

The component type *PipeMeasure* forms a measuring point in the pipe. You insert it with its connectors *A* and *B* at the desired measuring point in the flow network.

The following measured variables are output via connector *Measure*:

- Absolute value $|\dot{m}|$ of the flow
- Pressure p_A at connector *A*
- Temperature *T*

All other signals of connector *Measure* are not set in this component.

Note

The measuring procedures for the different variables are not simulated with suitable models. Only the quantities calculated in the flow network solver are output.

The direction of the flow is indicated by an arrow in the symbol during active simulation (see figure).



Figure 8-33 Media flow indicator of the component type PipeMeasure

Parameter

Parameter name	Description	Unit	Default value
geoHeight	Height of the measuring point	m	0

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure	p	bar
Mass flow	$\dot{m}$	kg/s
Temperature	T	°C

The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Enthalpy	h	kJ/kg
Density	r	kg/m³

See also

Geodetic height (Page 618)

PressureIndicator – Pressure indicator

Symbol



Figure 8-34 Symbol of component type PressureIndicator

Function

The component type *PressureIndicator* displays the pressure value specified using its *Measure* input.

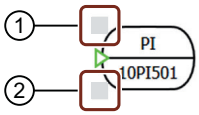
You can specify a high or low limit for the measured value with the *SwitchValue_Plus* or *SwitchValue_Minus* parameter. You can enable limit monitoring with the *Use_Plus* and *Use_Minus* switches. A green binary indicator in the symbol shows that the high or low switch value has been reached.

You can set the unit for the analog output and the switching limits with the *UoM* parameter.

For additional linking of measured values, you can make the following outputs of the component visible:

- *AOUT*
  Connection of the analog value with the unit set with parameter *UoM*
- *BOUT_Plus*
  Connection of the binary value of the high limit
- *BOUT_Minus*
  Connection of the binary value of the low limit

The state of these switch values is displayed dynamically in the graphic symbol:



① Display of the high limit switch value state
② Display of the low limit switch value state

Figure 8-35 Switch values in the graphic symbol of component type PressureIndicator

Parameters

Parameter name	Description	Unit	Default value
ICA_Code	MSR code, type of measurement	–	PI
Use_Plus	Activation of the high limit	see UoM	False
SwitchValue_Plus	Switch value for high limit	–	0.0
Use_Minus	Activation of the low limit	see UoM	False
SwitchValue_Minus	Switch value for low limit	–	0.0
UoM	Returns the measured value in the selected unit. Switch values are also in the selected unit. The following units are available: atm, bar, inHg, inWC, kPa, mbar, mmHg, MPa, mWS, Pa, psi, Torr		bar

Operating window

The following variable is displayed in the operating window:

Variable	Symbol	Unit
Pressure	p	bar mbar kPa Pa psi

TemperatureIndicator – Temperature indicator

Symbol



Figure 8-36 Symbol of component type TemperatureIndicator

Function

The component type *TemperatureIndicator* displays the temperature specified using its *Measure* input.

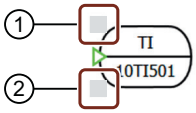
You can specify a high or low limit for the measured value with the *SwitchValue_Plus* or *SwitchValue_Minus* parameter. You can enable limit monitoring with the *Use_Plus* and *Use_Minus* switches. A green binary indicator in the symbol shows that the high or low switch value has been reached.

You can set the unit for the analog output and the switching limits with the *UoM* parameter.

For additional linking of measured values, you can make the following outputs of the component visible:

- *AOUT*
  Connection of the analog value with the unit set with parameter *UoM*
- *BOUT_Plus*
  Connection of the binary value of the high limit
- *BOUT_Minus*
  Connection of the binary value of the low limit

The state of these switch values is displayed dynamically in the graphic symbol:



① Display of the switch value state
② Display of the switch value state

Figure 8-37 Switch values in the graphic symbol of component type TemperatureIndicator

Parameters

Parameter name	Description	Unit	Default value
ICA_Code	MSR code, type of measurement	–	TI
Use_Plus	Activation of the high limit	–	False
SwitchValue_Plus	Switch value for high limit	–	0.0
Use_Minus	Activation of the low limit	–	False

Parameter name	Description	Unit	Default value
SwitchValue_Minus	Switch value for low limit	–	0.0
UoM	Returns the measured value in the selected unit. Switch values are also in the selected unit. The following units are available: °C, °F, K		°C

Operating window

The following variable is displayed in the operating window:

Variable	Symbol	Unit
Temperature	T	°C °F K

WeightIndicator – Weight indicator

Symbol



Figure 8-38 Symbol of component type WeightIndicator

Function

The *WeightIndicator* component type displays the weight specified using its *Measure* input.

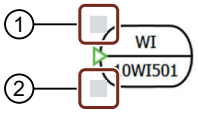
You can specify a high or low limit for the measured value with the *SwitchValue_Plus* or *SwitchValue_Minus* parameter. You can enable limit monitoring with the *Use_Plus* and *Use_Minus* switches. A green binary indicator in the symbol shows that the high or low switch value has been reached.

You can set the unit for the analog output and the switching limits with the *UoM* parameter.

For additional linking of measured values, you can make the following outputs of the component visible:

- *AOUT*
  Connection of the analog value with the unit set with parameter *UoM*
- *BOUT_Plus*
  Connection of the binary value of the high limit
- *BOUT_Minus*
  Connection of the binary value of the low limit

The state of these switch values is displayed dynamically in the graphic symbol:



① Display of the high limit switch value state
② Display of the low limit switch value state

Figure 8-39 Switch values in the graphic symbol of component type WeightIndicator

Parameters

Parameter name	Description	Unit	Default value
ICA_Code	MSR code, type of measurement	–	TI
Use_Plus	Activation of the high limit	–	False
SwitchValue_Plus	Switch value for high limit	–	0.0
UseMinus	Activation of the low limit	–	False
SwitchValue_Minus	Switch value for low limit	–	0.0
UoM	Returns the measured value in the selected unit. Switch values are also in the selected unit. The following units are available: g, kg, lb, t	–	kg

Operating window

The following variable is displayed in the operating window:

Variable	Symbol	Unit
Weight	m	kg

8.2.5.6 Mixing apparatuses

Mixer

Symbol

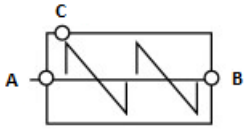


Figure 8-40 Symbol of component type Mixer

Function

The mixer is used to add a partial mass flow *C* (e.g. of a solid mass fraction or of a liquid mass fraction) to a main flow *AB*.

The mass fraction of the partial mass flow in the main flow is calculated in the mixer. This can be useful, for example, in a circulating circuit when a second substance is added (e.g. concentration of an acid).

The flow network of the partial mass flow *C* is not connected topologically to the flow network of the main mass flow *AB*. The change of the material properties (e.g. heat capacity) of the output flow due to addition of partial flow *C* is neglected.

If a solid is added, it is handled like a liquid. A pressure difference is therefore necessary for solid transport. The reason is that the flownet in SIMIT generally only permits the transport of gases and liquids. Any conveying equipment is replaced with equivalent connections (consisting of pumps, valves, sources and sinks).

The main mass flow can flow in both directions. The partial mass flow always flows into the mixer. The requirement for this is a higher pressure in the flownet of the added flow than in the main flow.

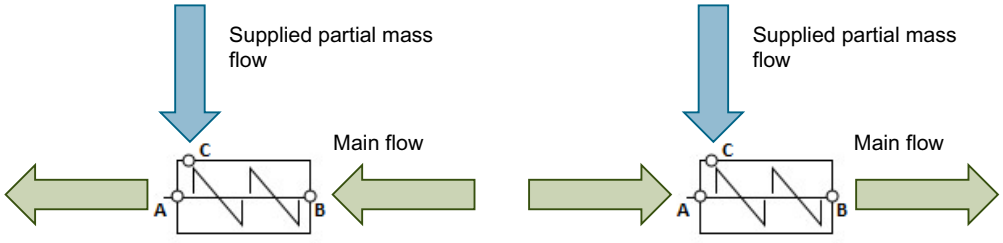


Figure 8-41 Possible flows in component type Mixer

The main flow connectors *A* and *B* are designed identically. The partial mass flow to be added is always fed through the connector *C*.

At each connector of the component *Mixer* the mass fraction of the added material can be specified by the inputs *RateA*, *RateB*, or *RateC*.

If none of the three connectors is specified (default setting), the following applies:

- 0 % mass fraction in the main flow entry
- 0 % mass fraction in the partial mass flow entry

The mass fraction of the total flow exiting at *A* or *B* is calculated and displayed at the output *Rate*.

A pressure drop is calculated for the main mass flow. This pressure drop depends on:

- Flow coefficient of the mixer (k_{VS} value)
- Throughflow (volume flow, medium)

The pressure drop above the mixer is calculated using the formula below:

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_{VS}^2 \; \rho \frac{\text{kg}}{\text{m}^3}} \; 12960 \left(\frac{\text{sec}}{\text{h}} \right)^2$$

Whereby:

$\Delta p = p_B - p_A$	The pressure drop across the mixer in bar
$\dot{m}$	The mass flow in kg/s
ρ	The density of the medium in kg/m³
k_{VS}	The flow coefficient of the mixer in m³/h

NOTICE
The flow network of the feed-in mass flow at connector *C* is terminated in the component with a mass flow source. Because of this mass flow source, the pressure reached at the edge of the flow network is dependent on the fed mass flow and the pressure drop in the feeding branch. Pressure drops that are too high in the feeding branch (for example too low k_{VS} values in the main flow) can result in impossible physical states.

Parameters

Parameter name	Description	Unit	Default value
Kvs	Flow coefficient of the mixer k_{VS} with $k_{VS} \geq 10^{-6}$ m³/h	m³/h	10.0
geoHeight	Geodetic height; can be changed online	m	0.0

Operating window

The following variables for the following mass flows are displayed in the operating window:

- Incoming main mass flow *A* or *B*
- Added mass flow *C*
- Outgoing total mass flow *A* or *B*

Variable	Symbol	Unit
Mass fraction	w	%
Mass flow	$\dot{m}$	kg/s

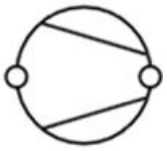
See also

Geodetic height (Page 618)

8.2.5.7 Pumps

Compressor

Symbol



Symbol of component type Compressor

Function

The *Compressor* component type calculates the pressure increase of a gas flow through a compressor as a function of the flow rate and speed using the following formula:

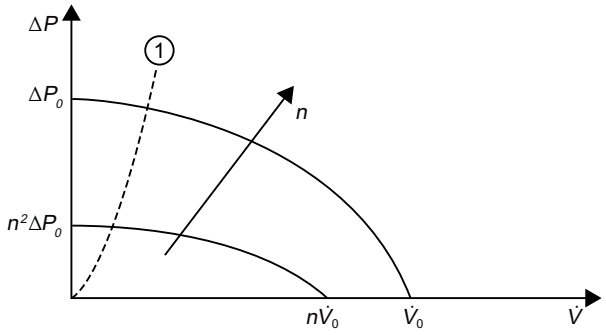
$$\Delta P = n^2 \Delta P_0 + \left(\Delta P^* - \Delta P_0\right) \frac{\dot{V}^2}{\left(\dot{V}^*\right)^2} \quad \text{for} \quad \dot{V} > 0$$

ΔP	Current pressure increase in the compressor [bar]
ΔP^*	Nominal pressure increase in the compressor at the design point [bar]
ΔP_0	Zero flow head of the compressor [bar]
n	Normalized current speed [0..1]
$\dot{v}$	Current flow rate in the compressor [Nm^3/s]
$\dot{v}^*$	Nominal flow rate in the compressor at the design point [Nm^3/s]

For the flow rate $\dot{v}$, the reference direction is defined from connector *A* to connector *B*; i.e. for a flow rate in the reference direction, $\dot{v} > 0$.

The speed is specified as a percentage value at the *Speed* input. The speed value is limited to the range $0 \le N \le 100$ %. This means n = N / 100% and thus $0 \le n \le 1$. This speed value can be linked, for example, over an I/O level as described in Coupling simulation model with actuator/sensor level (Page 631) to a control technology using the drives from the SIMIT basic library.

The following figure illustrates the quadratic relationship, defined above, between pressure increase and flow rate during compressor operation in the standard range, which means for $\dot{v} > 0$, $\Delta P > 0$.



① Compressor limit

Qualitative compressor characteristic

An polytropic process with the polytropic exponent *n* is assumed for the temperature increase resulting from compression of the gas:

$$T_B = T_A \left(\frac{P_B}{P_A}\right)^{\frac{n-1}{n}}$$

For an isothermal process, the polytropic exponent is: $n = 1$

For an isentropic process, the polytropic exponent is equal to the isentropic exponent *K*. Refer to the following table for the values of the isentropic exponent.

For a polytropic process, the polytropic exponent lies between the value for an isothermal and for an isentropic process.

Gas molecule	Isentropic exponents *K*
1-atom (e.g. helium, argon)	1.66
2-atom (e.g. O2, N2, air)	1.4
3-atom (e.g. water vapor)	1.3

There is no pressure increase following a flow reversal ($\dot{v} < 0$). The compressor behaves like a check valve in this case. You set the throttle effect in this case using the *Kv0* parameter.

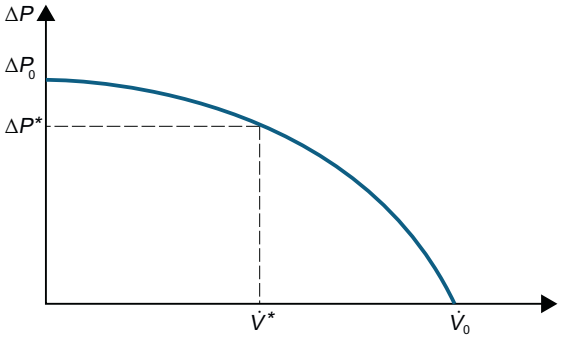
Note

The *Compressor* component only compresses gases. For liquids and water, the pump only acts as a flow resistor with the resistance coefficient *Kv0*. Water/steam is compressed with a liquid proportion of up to 5 Ma.%.

Parameters

The determining variables of the compressor characteristics are set with parameters:

Parameter name	Description	Unit	Default value
ZeroFlowHead	Zero flow head ΔP_0, $\Delta P_0 > \Delta P^*$	bar	5
NominalPressure	Nominal pressure ΔP^*; $\Delta P^* > 0$	bar	4
NominalVolumeflow	Nominal mass flow $\dot{v}^*$, $\dot{v}^* > 0$	Nm^3/s	1



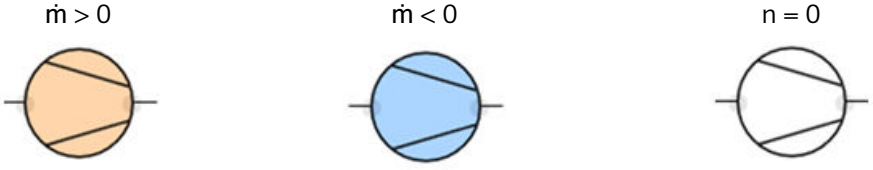
ΔP_0 Zero flow head [bar]
ΔP^* Nominal pressure [bar]
$\dot{v}^*$ Nominal volumetric flow [kg/s]

Graphic display of the parameters of component type *Compressor*

Additional parameters

Parameter name	Description	Unit	Default value
Kv0	Flow coefficient k_v	m^3/h	0.001
Pmin	Minimum pressure that can be generated at the suction side	bar	0.01
InitialEnth	Enthalpy of the gases upon initialization	kJ/kg	100
geoHeight	Geodetic height; can be changed online	m	0.0
PolytropicExponent	Polytropic exponent	-	1.4

The operating state in the compressor symbol is displayed as follows:



Symbol display of the compressor operating state

Operating window

You switch between automated control (control using the actuator-sensor level) and manual control in the operating window. The default is always automated control. The following variables are displayed in the operating window:

Variable	Symbol	Unit
Local control	–	–
"Slider"	–	%

The compressor characteristic curve is displayed graphically in the extended operating window of the *Compressor* component type. The following variables are also displayed:

Variable	Symbol	Unit
Pressure increase	ΔP	bar
Flow rate	$\dot{v}$	Nm^3/s
Nominal pressure	Ordinate	bar
Standard volumetric flow	Abscissa	Nm^3/s

See also

Geodetic height (Page 618)

LiquidRingPump

Symbol

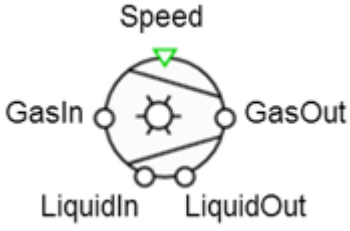


Figure 8-42 LiquidRingPump

Function

The component type *LiquidRingPump* calculates the pressure increase of a gas flow through a liquid ring pump as a function of the flow rate and speed using the following formula:

$$\Delta P = n^2 \Delta P_0 + (\Delta P^* - \Delta P_0) \frac{\dot{V}^2}{\dot{V}^{*2}} \quad \text{for } \dot{V} > 0$$

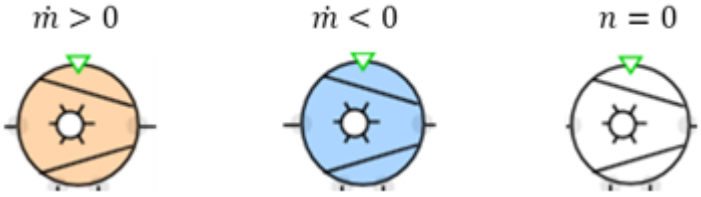
ΔP	Current pressure increase of the liquid ring pump (bar)
ΔP*	Nominal pressure increase of the liquid ring pump at the operating point (bar)
ΔP_0	Zero flow head of the liquid ring pump (bar)
n	Normalized current speed (0...1)
V̇	Current flow rate of the liquid ring pump (Nm^3/s)
V̇*	Nominal flow rate of the liquid ring pump at the operating point (Nm^3/s)

For the flow rate V̇, the reference direction is defined from connector *GasIn* to connector *GasOut*; i.e. for a flow rate in the reference direction, V̇ > 0.

The speed is specified as a percentage value at the *Speed* input. The speed value is limited to the range *0 ≤ N ≤ 100%*. This means n = N / 100% and thus *0 ≤ n ≤ 1*. This speed value can be connected, e.g. via an I/O level as described in "Coupling simulation model with actuator/sensor level" to a control technology using the drives from the SIMIT basic library.

Coupling simulation model with actuator/sensor level (Page 631)

The operating state in the symbol of the liquid ring pump is displayed as follows:



The following figure illustrates the above-defined quadratic relationship between pressure increase and flow rate during operation of the liquid ring pump in the normal range, i.e. for V̇ > 0, Δ*P* > 0.

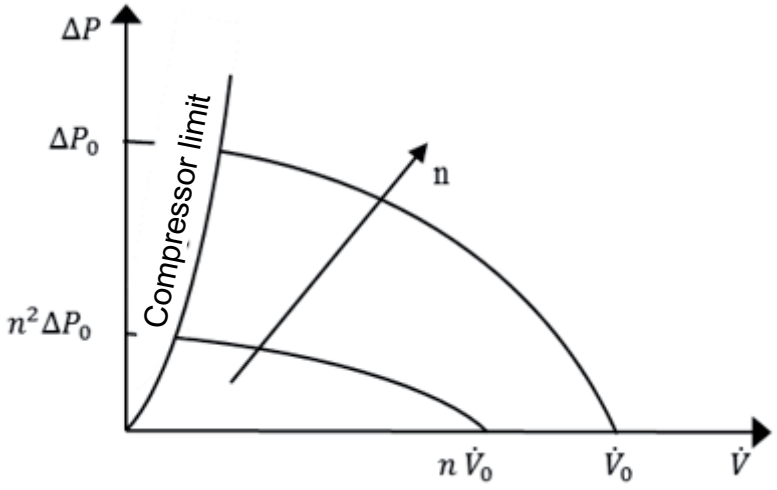


Figure 8-43 Qualitative fluid ring pump characteristic

In addition to the gas connector, the *LiquidRingPump* also has a connector for a liquid. All connectors must be connected because an open flow network would otherwise arise during the simulation. The two connectors for gas and liquid are independent of flow and pressure. They are only influenced by the heat transfer between two phases.

An isentropic process is assumed for the temperature increase resulting from compression of the gas:

$$T_B = T_A \left(\frac{P_B}{P_A}\right)^{\frac{\kappa-1}{\kappa}}$$

Heat exchange with the liquid also occurs. The *HeatTransferCoefficient* parameter can be used to influence the heat transfer. The parameter is the product of heat transfer coefficients and surface area.

There is no pressure increase in the case of flow reversal of the gas flow ($\dot{V} < 0$). The liquid ring pump behaves like a check valve for the gas flow in this case. The throttle effect is set using the *Kv0* parameter.

Parameters

The determining variables of the pump characteristic are set with parameters:

Parameter name	Description	Unit	Default value
ZeroFlowHead	Zero flow head ΔP_0, $\Delta P_0 > \Delta P^*$	bar	5
NominalPressure	Nominal pressure ΔP^*; $\Delta P^* > 0$	bar	4
NominalVolumeflow	Nominal flow rate *, * > 0	Nm^3/s	1

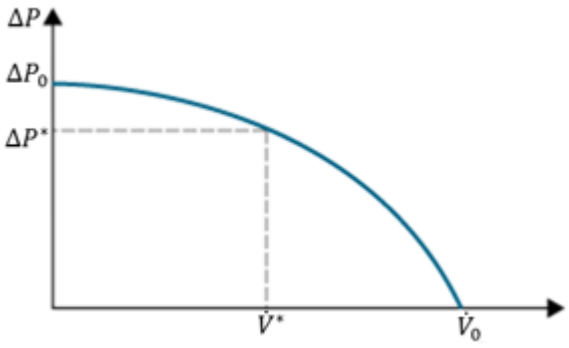


Figure 8-44 Graphic representation of the parameters of LiquidRingPump

Additional parameters

Parameter name	Description	Unit	Default value
KvLiquid	Flow coefficient for the liquid	m^3/h	10.0
HeatTransferCoefficient	Product of heat transfer coefficient and surface area for heat transfer of gas and liquid phases	kW/K	5.0
Pmin	Minimum pressure that can be generated at the suction side	bar	0.01
Kv0	Flow coefficient k_v in the case of back flow	m^3/h	0.001
TemperatureInit	Temperature upon initialization	C	100.0
geoHeight	Geodetic height; can be changed online	m	0.0

Operating window

You switch between automated control strategy (control using the actuator-sensor level) and manual control strategy in the operating window. The automated control strategy is always preset. The following variables are displayed in the operating window:

Variable	Symbol	Unit
Local control	-	-
Slider	-	%

PistonCompressor

Symbol

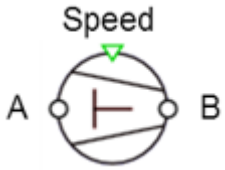


Figure 8-45 Symbol of component type PistonCompressor

Function

The component type *PistonCompressor* pumps a gas volume flow from connector *A* to connector *B* as a function of speed. Liquid is not pumped. In the case of a piston compressor, the volume flow is not dependent on the pressure. The volume flow is pumped from the compressor independent of the pressure on the pressure side.

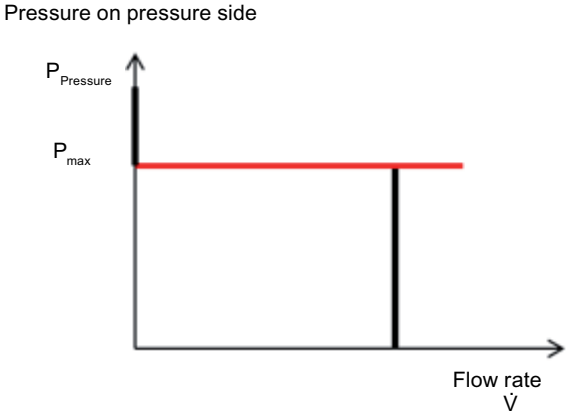


Figure 8-46 Compressor characteristic

To prevent mechanical damage to the compressor from excessive pressures on the pressure side (P_{Druck}), an overflow protection is available. When the maximum pressure *Pmax* is exceeded, the overflow protection opens and a portion of the flow is diverted from the pressure side to the suction side. The overflow protection is controlled in such a way that the maximum permissible pressure on the pressure side is adjusted. When the overpressure protection is tripped, higher pressures than the maximum pressure setting can occur briefly.

The speed (0 - 100%) is specified at the *Speed* input. The nominal volume flow *nVfl* is achieved at maximum pump speed (100%).

The control of the overflow protection can be dampened with the damping factor *dump* .

If the compressor is switched off (*Speed = 0*) and the pressure on the pressure side is greater than the response pressure of the overflow protection *Pmax*, a flow across the overflow protection opposite the actual flow direction from connector *A* to connector *B* may occur ($\dot{m} < 0$).

The operating state of the compressor is displayed as follows:

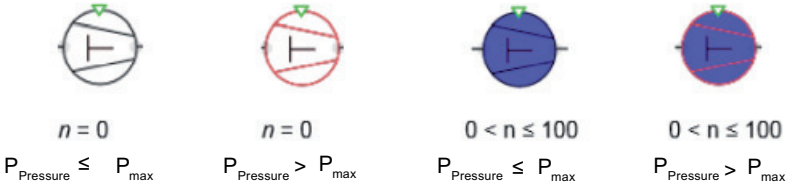


Figure 8-47 Operating state of the compressor

Parameters

Parameter name	Description	Unit	Default value
nVfl	Nominal volume flow	m^3/s	1.0
pmax	Maximum pressure on suction side	bar	10

Additional parameters

Parameter name	Description	Unit	Default value
dump	Damping factor of overflow controller	-	1

PistonPump

Symbol

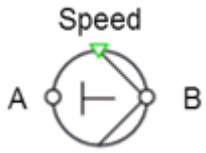


Figure 8-48 PistonPump

Function

The component type *PistonPump* pumps a liquid volume flow from connector *A* to connector *B* as a function of speed. Gas is not pumped. In the case of a displacement pump, the volume flow is not dependent on the pressures. The volume flow is pumped from the pump independent of the pressure on the pressure side.

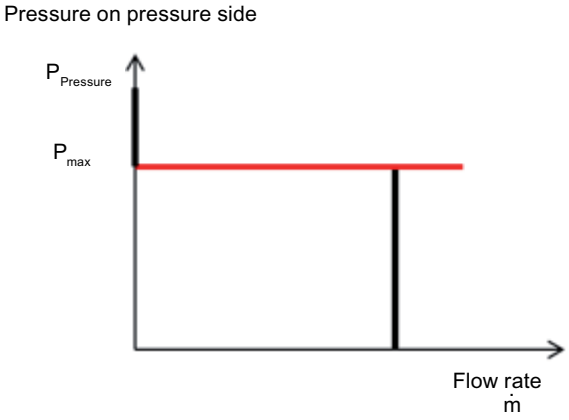


Figure 8-49 Pump characteristic

To prevent mechanical damage to the pump from excessive pressures on the pressure side ($_{(}P_{Druck}$), an overflow protection is available. When the maximum pressure P_{max} is exceeded, the overflow protection opens and a portion of the flow is diverted from the pressure side to the suction side. The overflow protection is controlled in such a way that the maximum permissible pressure on the pressure side is adjusted. When the overpressure protection is tripped, higher pressures than the maximum pressure setting can occur briefly.

The speed (0 - 100%) is specified at the *Speed* input. The nominal volume flow *nVfl* is achieved at maximum pump speed (100%).

The control of the overflow protection can be dampened with the damping factor *dump* .

If the pump is switched off (Speed = 0) and the pressure on the pressure side is greater than the response pressure of the overflow protection P_{max}, a flow across the overflow protection opposite the actual flow direction from connector *A* to connector *B* may occur ($\dot{m} < 0$).

The operating state in the pump is displayed as follows (see figure):

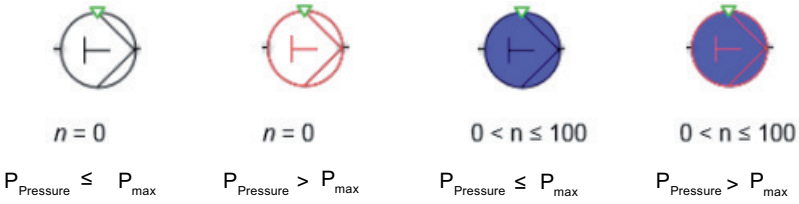


Figure 8-50 Operating status of the pump

Parameters

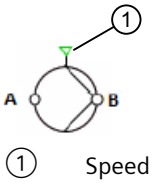
Parameter name	Description	Unit	Default value
nVfl	Nominal volume flow	m^3/s	0.001
pmax	Maximum pressure on suction side	bar	5

Additional parameters

Parameter name	Description	Unit	Default value
dump	Damping factor of overflow controller	-	1

Pump

Symbol



Speed

Figure 8-51 Symbol of component type Pump

Function

The component type *Pump* calculates the pressure increase through a pump as a function of flow rate and speed according to the following formula:

$$\Delta p = n^2 \Delta p_0 + \left(\Delta p^* - \Delta p_0\right) \frac{\dot{m}^2}{\left(\dot{m}^*\right)^2} \quad \text{for} \quad \dot{m} > 0$$

Δp	Current pressure increase of the pump [bar]
Δp^*	Nominal pressure increase of the pump at the design point [bar]
Δp_0	Zero flow head of the pump [bar]
n	Normalized current speed [0..1]
$\dot{m}$	Current flow rate of the pump [kg/s]
$\dot{m}^*$	Nominal flow rate of the pump at the design point [bar]

For the flow rate $\dot{m}$, the reference direction is defined from connector *A* to connector *B*; for a flow rate in reference direction, $\dot{m} > 0$.

The speed is defined as percentage at the *Speed* input. The speed value is limited to the range $0 \le N \le 100$ %. This means n = N / 100% and thus $0 \le n \le 1$. This speed value can, for example, be connected over its I/O level as described in Coupling simulation model with actuator/sensor level (Page 631) to a control technology using the drives from the SIMIT basic library.

The following figure illustrates the above-defined quadratic relationship between pressure increase and flow rate during pump operation in the standard range, which means for $\dot{m} > 0$, $\Delta p > 0$, in the figure below.

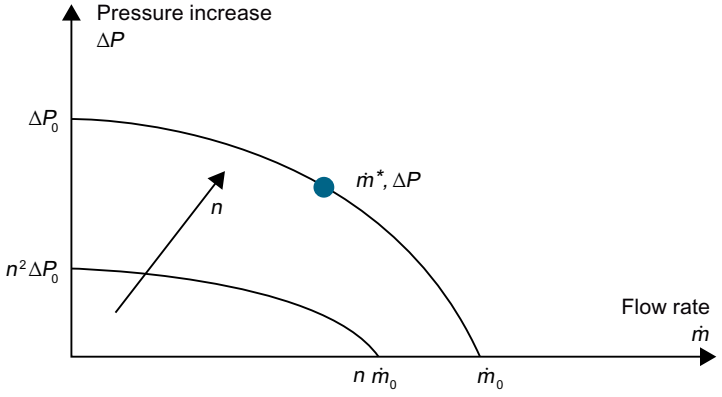


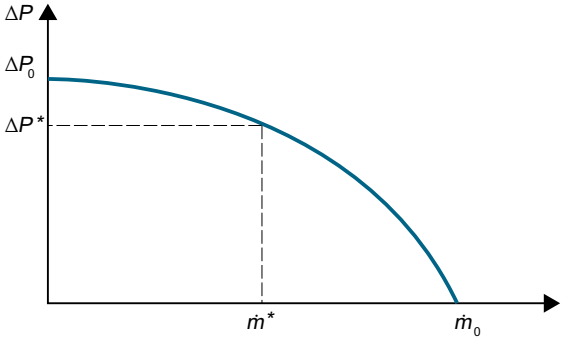
Figure 8-52 Qualitative pump characteristic

There is no pressure increase following a flow reversal ($\dot{m} < 0$). The pump behaves like a check valve in this case. You set the throttle effect in this case using parameter *Kv0*.

Parameters

Parameter name	Description	Unit	Default value
ZeroFlowHead	Zero flow head Δp_0, $\Delta p_0 > \Delta p^*$; can be changed online	bar	5.0
NominalPressure	Nominal pressure Δp^*; $\Delta p^* > 0$; modifiable online	bar	4.0
NominalMassflow	Nominal mass flow $\dot{m}^*$, $\dot{m}^* > 0$; modifiable online	kg/s	1.0

The determining variables of the pump characteristic are set with parameters:



Δp_0 Zero flow head [bar]

Δp^* Nominal pressure [bar]

$\dot{m}^*$ Rated mass flow [kg/s]

Figure 8-53 Graphic display of the parameters of component type Pump

Additional parameters

Parameter name	Description	Unit	Default value
Kv0	Pump throttle effect if flow is reversed	m^3/h	0.001
geoHeight	Geodetic height; can be changed online	m	0.0

The operating state in the pump symbol is displayed as follows (see figure):

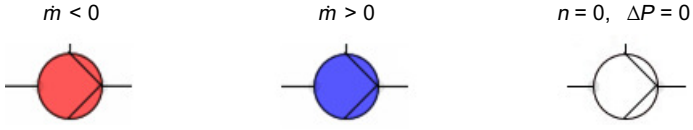


Figure 8-54 Display of the operating state of the pump at the symbol

Operating window

You switch between automated control (control using the actuator-sensor level) and manual control in the operating window. The default is always automated control.

For manual control, you specify the pump speed as a percentage using a slider.

The pump characteristic is displayed graphically in the extended operating window. The characteristic values of the characteristic curve and the current operating point are indicated via digital displays. In addition, the current pressure increase Δp and the flow rate $\dot{m}$ are displayed.

See also

Geodetic height (Page 618)

SteamJetPump

Symbol

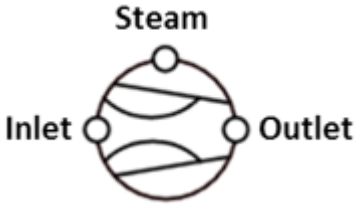


Figure 8-55 SteamJetPump

Function

The component type *SteamJetPump* calculates the pressure increase of a gas flow through a steam jet pump as a function of the steam pressure, steam quantity and gas quantity. For the flow rate $\dot{m}$, the reference direction is defined from connector *Inlet* to connector *Outlet*; for a flow rate in reference direction, $\dot{m}_{Gas} > 0$. The following pump characteristic qualitatively describes the relationship between steam pressure (*PSteam*), suction pressure (*PInlet*) and specific steam flow ($\dot{m}_{Steam}$ / $\dot{m}_{Gas}$).

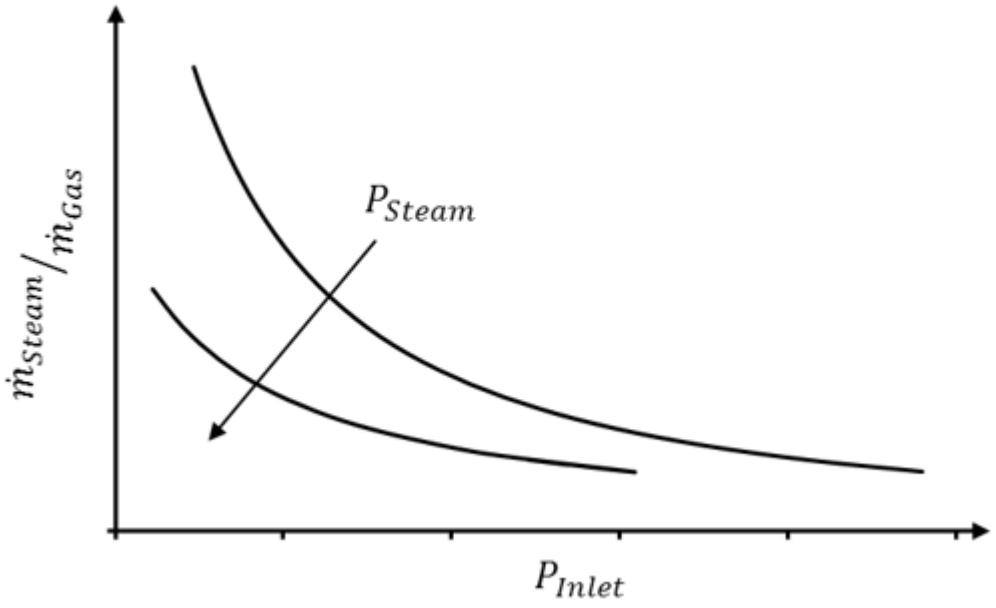


Figure 8-56 Qualitative SteamJetPump characteristic

Note

The *SteamJetPump* component only compresses gases and steam. For liquids and water, the *SteamJetPump* only acts as a flow resistance with resistance coefficient *Kv0*.

There is no pressure increase in the case of a flow reversal ($\dot{m}_{Gas} < 0$). The *SteamJetPump* behaves like a check valve in this case. You set the throttle effect in this case using the *Kv0* parameter.

All inputs, *Inlet*, *Outlet* and *Steam*, must be interconnected and the adjoining flow networks must be defined using the *NetParam* component. None of these inputs may remain open during simulation because this would create open flow networks. This would, in turn, result in an error message during code generation.

The operating state in the symbol of the *SteamJetPump* is displayed as follows:

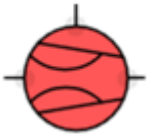


Figure 8-57 Error status



Figure 8-58 Normal operation

The symbol is blue during normal operation. If back flows occur due to the given operating conditions, the symbol turns red to indicate the error state.

The *SteamJetPump* can be configured using the resistance coefficient *KvSteam* on the steam side and the resistance coefficient *KvGas* on the gas side. The *Variability* parameter can be used to adapt the rate of change of the pressure adjustment to changed operating conditions. If values are too low, the state of equilibrium may be reached only very slowly or may not be reached at all. If values are too high, instability may result.

Parameters

Parameter name	Description	Unit	Default value
KvSteam	Resistance coefficient on steam side	m^3/h	10.0
KvGas	Resistance coefficient on gas side	m^3/h	500.0

Additional parameters

Parameter name	Description	Unit	Default value
TemperatureInit	Initialization temperature	°C	20.0
PressureInit	Initialization pressure	bar	0.01
HeatCapacityInit	Initialization heat capacity	kJ/kgK	4.18
Variability	Variability of the pressure change	-	1.0
Kv0	Throttle effect if flow is reversed	m^3/h	0.000001

Operating window

Variable	Symbol	Unit
Mass flow of steam	MFL_{Steam}	kg/s
Mass flow of gas	MFL_{Gas}	kg/s
Suction pressure	$P_{Suction}$	mbar

Turbine

Symbol

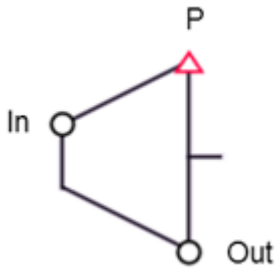


Figure 8-59 Turbine

Function

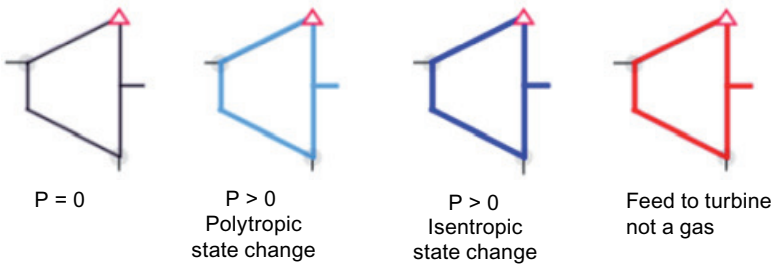
The component type *Turbine* calculates the usable power from expansion of a gas flow. A polytropic process is assumed for the expansion. The turbine is configured by specifying a stationary operating point (*OP*):

- Gas flow *OP_Mfl*
- Power output *OP_Power*
- Pressure at turbine inlet *OP_PressureIn*
- Pressure at outlet *OP_PressureOut*
- Temperature at inlet *OP_TemperatureIn*

In the normal state, the turbine symbol is black (no power output) or light blue (power output). The dark blue symbol indicates when the polytropic exponent is unable to be determined due to the specified operating point. In this case, an isentropic process is assumed. If the symbol is red, the medium at the turbine inlet is not a gas. In the case of water/steam, a warning is also output if condensation will occur during the expansion process.

From this data, a determination is made of the polytropic exponent and the pressure drop coefficient for the gas flow in the reference direction from connector In to connector *Out*. For back flows, the turbine behaves like a check valve with pressure drop coefficient *Kv0*.

The various operating states of the turbine are displayed in the symbol as follows:



Parameters

Parameter name	Description	Unit	Default value
OP_Mfl	Gas flow at the operating point	kg/h	1.0
OP_Power	Power output at the operating point	kW	200.0
OP_PressureIn	Pressure at the inlet of the turbine at the operating point	bar	15
OP_PressureIn	Pressure at the outlet of the turbine at the operating point	bar	5
OP_TemperatureIn	Inlet temperature of the gas flow to the turbine at the operating point	°C	400

Additional parameters

Parameter name	Description	Unit	Default value
Efficiency	Efficiency of the turbine	%	100
Kv0	Throttle effect if flow is reversed	m^3/h	0.000001
geoHeight	Geodetic height	M	0

Operating window

Variable	Symbol	Unit
Power	P	kW
Pressure difference across the turbine	ΔP	bar
Outlet temperature	T_{out}	°C

8.2.5.8 Reactors

MergeComponentVector

Symbol

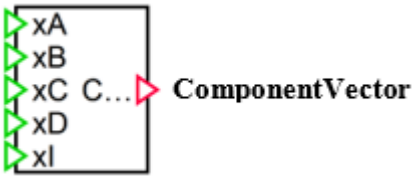


Figure 8-60 MergeComponentVector

Function

The component *MergeComponentVector* writes 5 analog input values to an output vector.

This output vector can be linked, for example, with an input of the *TankReactor* or *TubeReactor* component.

The inputs specify mass fractions (kg/kg). Values from *0 – 1* are allowed. The sum of all mass fractions should be *1*. If the sum is not 1, the mass fractions are normalized so that the sum is *1*.

SeparateComponentVector

Symbol

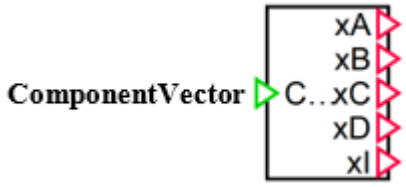


Figure 8-61 SeparateComponentVector

Function

The *SeparateComponentVector* component writes an input vector to 5 analog output values.

This input vector can be linked, for example, with an output of the *TankReactor* or *TubeReactor* component.

The inputs specify mass fractions (kg/kg). Values from *0 – 1* are allowed. The sum of all mass fractions should be *1*. If the sum is not *1*, the mass fractions are normalized so that the sum is *1*.

TankReactor

Symbol

Function

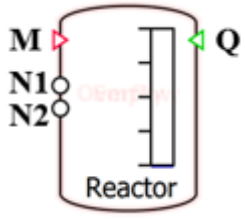


Figure 8-62 TankReactor

The *TankReactor* component calculates the simulation of a chemical reaction. Two reactants, components A and B, are transformed into one or two products, components C and D. An inert component I can also be present. The reaction type is set with the *ReactionType* parameter. The reaction can proceed according to one of the following reaction types:

- A + B > C
- A + B > 2C
- A + 2B > C
- A + 2B > 2C
- A + B > C + D
- A + B > C + 2D
- A + 2B > C + D
- A + 2B > C + 2D

The reaction can proceed in the gas phase or liquid phase. The reaction mass therefore corresponds to the mass of the gas phase or liquid phase. A back reaction from the products to the reactants is also possible (balance reaction). This is set using the ReactionConstant2 parameter (0 = No back reaction). A simplifying assumption is made that all reactants have the same molar mass. The same applies to the products. However, reactants and products can have different molar masses. This is specified using the stoichiometry (reaction type). For example, when A + 2B > C, it follows that C has three times the molar mass of the products (A or B).

The reaction kinetics is represented with the exponential equation. In so doing, the order of reaction with respect to every component is 1:

$$r = k_1 \cdot x_A \cdot x_B - k_2 \cdot x_C \cdot x_D$$

The fractions of reactants and products are specified as mass fraction x_i in weight percent. The temperature dependency of the reaction kinetics is calculated with the Arrhenius equation:

$$k_i = ReactionConstant_i \cdot \exp\left(-\frac{Ea_i}{\mathrm{R} \cdot (Temperture + 273{,}15)}\right)$$

where R is the general gas constant and *Temperature* is the reaction temperature in °C. The reaction temperature results from the enthalpy balance of the fed and removed flows, the supplied or discharged heat quantity via connector *Q* and the enthalpy of reaction of an endothermic or exothermic reaction.

The *CatalystActivity* input can be used to specify the activity of a catalyst (in %). The value can vary between 0% (no reaction) and 100%. The reaction rate r is adapted accordingly.

$$r_{\text{effective}} = CatalystActivity \cdot r$$

CatalystActivity can be used, for example, to simulate a catalyst deactivation or a loss of catalyst mass.

Interconnection

The *N1 ..Nn* connectors are the connection points for the material flows, which means for the connection of pipes. The number of these connection points depends on the system to be simulated and is specified with the *NbrOfConnectors* parameter. All connectors of the input vector *N1...Nn* are initially equivalent and can be used for all connector types (water/steam, ideal gas, liquid). For each entering mass flow, the mass concentration (kg/kg) must be specified for each component (A, B, C, D, I). Exiting mass flows have the concentration of the reaction mixture. To merge the 5 analog values for the mass concentration into an input vector, the *MergeComponentVector* component is available. To separate the output vector into 5 analog values again, the *SeparateComponentVector* component is available.

The medium in the adjoining flow networks is defined using the *NetParam* component. Water/steam is treated as an ideal gas or as a liquid, depending on the temperature and pressure. The enthalpy of vaporization of water is disregarded

You must interconnect all defined inputs in the input vector *N1 ..Nn*. None of these inputs may remain open during simulation because this would create open flow networks. This would, in turn, result in an error message during code generation.

Parameters

Parameter name	Description	Unit	Default value
Volume	Volume of tank	m^3	1.0
Height	Height of reactor	m	1.0
ReaktionConstant1	Reaction constant for the reaction	kg/m^3s	30
ReaktionConstant2	Reaction constant for the back reaction	kg/m^3s	0
ReactionType	Reaction type	-	A + B -> C
NbrOfConnectors	Number N of connectors	-	2
ConnectorHeight	Height of the pipe connectors above the tank bottom	m	0.0
NbrOfMeasurements	Number N of process tags	-	1
MeasurementHeight	Height of the process tags above the tank bottom	m	0.0

Additional parameters

Parameter name	Description	Unit	Default value
Ea1	Activation energy of the reaction	kJ/kg	30.0
Ea2	Activation energy of the back reaction	kJ/kg	30.0
TemperatureInit	Initialization temperature	°C	20.0
PressureInit	Initialization pressure	bar	1.0
LevelInit	Initialization level of the reactor	%	20.0

xA_Init	Initialization fraction of component A of the reaction mass	kg/kg	0.0
xB_Init	Initialization fraction of component B of the reaction mass	kg/kg	0.0
xC_Init	Initialization fraction of component C of the reaction mass	kg/kg	0.0
xD_Init	Initialization fraction of component D of the reaction mass	kg/kg	0.0
xI_Init	Initialization fraction of component I of the reaction mass	kg/kg	1.0
InitCpLiquid	Initialization heat capacity of the liquid phase	kJ/kgK	4.18
InitCpGas	Initialization heat capacity of the gas phase	kJ/kgK	1.0
Density	Liquid density in the tank	kg/m³	997.337
MinGasVolumePercent	Percentage of tank volume that cannot be filled with liquid when filling the tank	%	5.0
MinLiquidVolumePercent	Percentage of tank volume that remains in the tank when draining the tank	%	0.5
geoHeight	Geodetic height of the tank	m	0.0

Operating window

Variable	Symbol	Unit
Temperature	T	°C
Pressure	P	Bar
Reaction mass	$m_{R'}$	kg
Total mass	m_{tot}	kg
Level	L	m
Mass fraction of component A	x_A	kg/kg %
Mass fraction of component B	x_B	kg/kg %
Mass fraction of component C	x_C	kg/kg %
Mass fraction of component D	x_D	kg/kg %
Mass fraction of component I	x_I	kg/kg %
Reaction type	ReactionType	-
Reaction phase	ReactionPhase	-

TubeReactor

Symbol

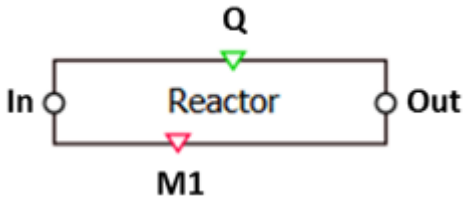


Figure 8-63 TubeReactor

Function

The *TubeReactor* component calculates the simulation of a chemical reaction in a tube reactor with plug flow. Two reactants, components A and B, are transformed into one or two products, components C and D. An inert component I can also be present. The reaction type is set with the *ReactionType* parameter. The reaction can proceed according to one of the following reaction types:

- A + B > C
- A + B > 2C
- A + 2B > C
- A + 2B > 2C
- A + B > C + D
- A + B > C + 2D
- A + 2B > C + D
- A + 2B > C + 2D

The reaction is a gas phase reaction or liquid phase reaction depending on the network connector. A back reaction from the products to the reactants is also possible (balance reaction). This is set using the ReactionConstant2 parameter (0 = No back reaction). A simplifying assumption is made that all reactants have the same molar mass. The same applies to the products. However, reactants and products can have different molar masses. This is specified using the stoichiometry (reaction type). For example, when A + 2B > C, it follows that C has three times the molar mass of the products (A, B).

The reaction kinetics is represented with the exponential equation. In so doing, the order of reaction with respect to every component is 1:

$$r = k_1 \cdot x_A \cdot x_B - k_2 \cdot x_C \cdot x_D$$

The fractions of reactants and products are specified as mass fraction x_i in weight percent. The temperature dependency of the reaction kinetics is calculated with the Arrhenius equation:

$$k_i = ReactionConstant_i \cdot \exp\left(-\frac{Ea_i}{\mathrm{R} \cdot (Temperture + 273{,}15)}\right)$$

where R is the general gas constant and *Temperature* is the reaction temperature in °C. The reaction temperature results from the enthalpy balance of the fed and removed flows, the supplied or discharged heat quantity via connector *Q* and the enthalpy of reaction of an endothermic or exothermic reaction.

The *CatalystActivity* input can be used to specify the activity of a catalyst (in %). The value can vary between 0% (no reaction) and 100%. The reaction rate r is adapted accordingly.

$$r_{\text{effective}} = CatalystActivity \cdot r$$

CatalystActivity can be used, for example, to simulate a catalyst deactivation or a loss of catalyst mass.

The *TubeDiscretization* is used to divide the tube into the required number of sections. A greater discretization usually produces more accurate results but also causes an increased CPU load. A high discretization can also lead to the situation that the time slice is too long to allow a complete calculation. This is indicated by a red outline around the reactor symbol. The discretization should be reduced in this case.



Interconnection

The medium flows into the reactor via connector In. The mass concentration (kg/kg) must be specified for each component (A, B, C, D, I). The medium flows out of the reactor via connector Out. The exiting mass flow has the concentration of the reaction mixture at the output. To merge the 5 analog values for the mass concentration into an input vector, the *MergeComponentVector* component is available. To separate the output vector into 5 analog values again, the *SeparateComponentVector* component is available.

The medium in the adjoining flow networks is defined using the *NetParam* component. Water/steam is treated as an ideal gas or as a liquid, depending on the temperature and pressure at the inlet of the reactor. The enthalpy of vaporization of water is disregarded

Parameters

Parameter name	Description	Unit	Default value
Length	Length of tube reactor	m	5
Diameter	Diameter of tube reactor	m	0.2
ReaktionConstant1	Reaction constant for the reaction	kg/m³s	30
ReaktionConstant2	Reaction constant for the back reaction	kg/m³s	0
ReactionType	Reaction type	-	A + B -> C

NbrOfMeasurements	Number N of measuring points	-	1
MeasurementPosition	Position of process tags (0 = Reactor inlet)	m	0.0

Additional parameters

Parameter name	Description	Unit	Default value
Ea1	Activation energy of the reaction	kJ/kg	30.0
Ea2	Activation energy of the back reaction	kJ/kg	30.0
TemperatureInit	Initialization temperature	°C	20.0
xA_Init	Initialization fraction of component A of the reaction mass	kg/kg	0.0
xB_Init	Initialization fraction of component B of the reaction mass	kg/kg	0.0
xC_Init	Initialization fraction of component C of the reaction mass	kg/kg	0.0
xD_Init	Initialization fraction of component D of the reaction mass	kg/kg	0.0
xI_Init	Initialization fraction of component I of the reaction mass	kg/kg	1.0
geoHeight	Geodetic height of the reactor	m	0.0
TubeDiscretization	Axial discretization of the tube reactor	-	10
Kvs	Flow coefficient kvs	m^3/h	5

Operating window

Variable	Symbol	Unit
Temperature	T	°C
Pressure	P	Bar
Mass fraction of component A	x_A	kg/kg %
Mass fraction of component B	x_B	kg/kg %
Mass fraction of component C	x_C	kg/kg %
Mass fraction of component D	x_D	kg/kg %
Mass fraction of component I	x_I	kg/kg %
Reaction type	ReactionType	-
Reaction phase	ReactionPhase	-

8.2.5.9 Separators

AbsorptionTower

Symbol

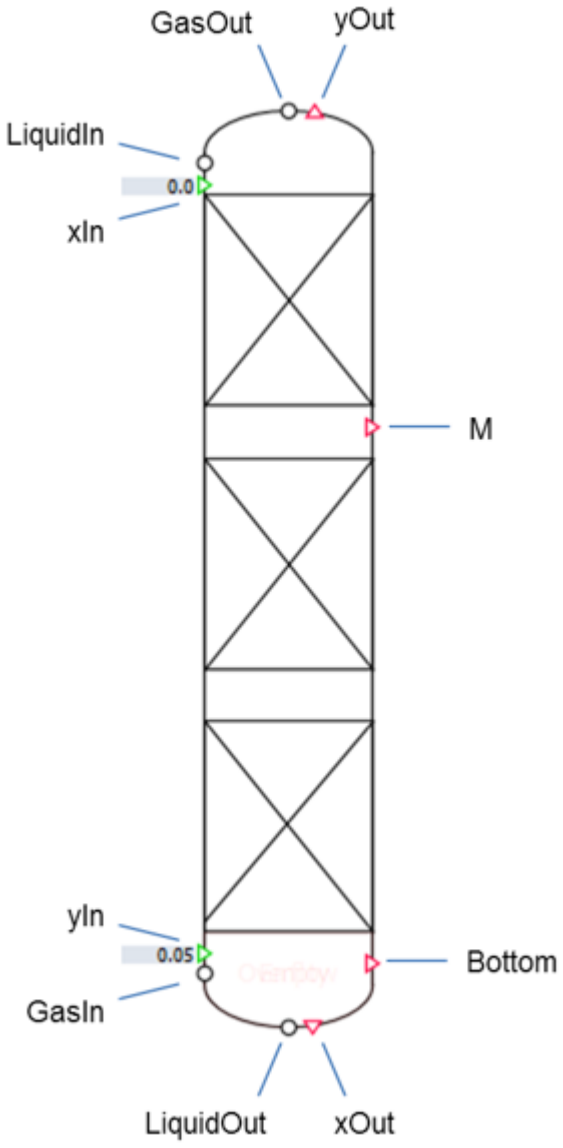


Figure 8-64 AbsorptionTower

Function

The component type *AbsorptionTower* uses a simplified calculation to calculate the absorption/desorption of an absorbate. In the case of absorption, the absorbate changes from the gas phase to the liquid phase. In the case of desorption, this is reversed.

The liquid flow is fed to the column via connector *LiquidIn* with the concentration of absorbate *xIn*. The liquid flow exits the column via connector *LiquidOut* with the concentration of absorbate *xOut*. The gas flow is fed to the column via connector *GasIn* with the concentration of absorbate *yIn*. The gas flow exits the column via connector *GasOut* with the concentration of absorbate *yOut*.

The absorption/desorption of the absorbate is calculated using the simplified Henry's Law:

$yP = Hx$

Here, y is the concentration of the absorbate in the gas phase, *x* the concentration in the liquid phase, *P* the pressure and *H* the Henry's coefficient. The Henry's coefficient is determined from the specifications for the (stationary) operating point (OP). The stationary operating point of the column is specified with the following parameters:

- Liquid inlet mass flow *OP_mflLiquidIn*
- Liquid inlet concentration *OP_xLiquidIn*
- Gas inlet mass flow *OP_mflGasIn*
- Gas inlet concentration *OP_yGasIn*
- Gas outlet concentration *OP_yGasOut*
- Absorption temperature at operating point *OP_Temperature*
- Absorption pressure at operating point *OP_Pressure*

The *TemperatureDependency* parameter can be used to specify the temperature dependency of the Henry's coefficient with the following equation:

$$H(T) = \frac{TemperatureDependency}{1000\,{}^{mbar}/_{bar}} \cdot (T - OP_Temperature) + H(OP_Temperature)$$

The column size is specified with the following parameters:

- Diameter of column *Diameter*
- Number of trays *NbrOfTrays*
- Distance between trays *TrayHeight*

The level, pressure and temperature in the column sump can be read out using the measuring point connection *Bottom*. Measurements (pressure and temperature) in the column can be read out using measurement point connector *Measurement[i]*. The *NbrOfMeasurements* parameter can be used to specify the number. The tray on which the measurement point is located can be specified for each measurement point using the *MeasurementsTray* parameter.

Parameters

Parameter name	Description	Unit	Default value
Diameter	Column diameter	m	2
NbrOfTrays	Number of trays	-	10
TrayHeight	Distance between trays	mm	500
NbrOfMeasurements	Number of measurement points of the column	-	1
MeasurementTray	Tray on which the measurement point is located	-	1
OP_mflLiquidIn	Liquid phase inlet flow at operating point	kg/s	50
OP_xLiquidIn	Concentration of absorbate in liquid inlet flow at operating point	kg/kg	0.0
OP_mflGasIn	Gas phase inlet flow at operating point	kg/kg	20.0
OP_yGasIn	Concentration of absorbate in gas inlet flow at operating point	kg/kg	0.05
OP_yGasOut	Concentration of absorbate in gas outlet flow at operating point	kg/s	0.001
OP_Temperature	Absorption temperature at operating point	°C	20.0
OP_Pressure	Pressure in the column at operating point	°C	1.0

Additional parameters

Parameter name	Description	Unit	Default value
TemperatureDependency	Temperature dependency of the Henry's coefficient	mbar/°C	0.0
Holdup	Liquid volume percentage of column volume	Vol.%	30.0
EnthalpyOfSolubility	Enthalpy of solution for dissolution of absorbate in the liquid phase	kJ/kg	0.0
PressureInit	Initialization pressure in the column	bar	1.0
TemperatureInit	Initialization temperature in the column	°C	20.0
LevelBottomInit	Initialization level of column sump	%	50
KvsColumn	Resistance coefficient for flow through the column	m^3/h	10000.0
geoHeight	Geodetic height of the column (at the sump)	m	0

Operating window

The main parameters of the column are displayed in the operating window.

Variable	Symbol	Unit
Top temperature	T	°C
Top pressure	P	bar
Liquid phase inlet flow	-	kg/s
Concentration in liquid phase inlet flow	-	kg/kg
Gas phase inlet flow	-	kg/s
Concentration in gas phase inlet flow	-	kg/kg
Liquid phase outlet flow	-	kg/s
Concentration in liquid phase outlet flow	-	kg/kg
Gas phase outlet flow	-	kg/s
Concentration in gas phase outlet flow	-	kg/kg
Sump temperature	T	°C
Sump pressure	P	bar
Pressure drop	ΔP	mbar

Centrifuge

Symbol

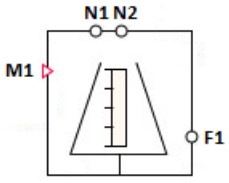


Figure 8-65 Symbol of component type Centrifuge

Function

The component type *Centrifuge* is used to separate a mixture (suspension or emulsion) consisting of two substances into its individual components. This component cannot be used to separate gas mixtures.

Separation in a centrifuge is the result of inertia during operation of the centrifuge (rotation of the centrifuge chamber). Particles with greater density are transported to the outside or solids are retained due to internals (e.g. screen or filter cloth).

In this model, the separation is simulated by defining a separable mixture component and a non-separable mixture component. The sum of both fractions is 100%. Different densities are ignored during the separation.

The substance which is transported to the outside is referred to as filtrate (separable mixture component). The substance that remains in the centrifuge chamber is referred to as filter cake below (non-separable mixture component).

The filter cake can be dissolved and washed out by adding water. By adding pressurized gas, the liquid can be forced out of the filter cake.

Circuit

The centrifuge has two types of connectors for material flows and thus for connection of pipes:

- **N1 ... Nn – general connecting pipes**
  These connection points are used to fill and empty the centrifuge and as gas connections. All connectors of the vector *N1..Nn* are initially equal and can be used for all connection types (water/steam, ideal gas, liquid).
- **F1 ... Fn – Filtrate drain**
  These connection points are a special case. The filtrate exits the system through these connections. All other connections are made using the vector *N1..Nn* described above.

The number of connection points depends on the system to be simulated and is specified with the parameters *NbrOfInletsOutlets* for the connectors *N1 ... Nn* and *NbrOfFiltrateOutlets* for the connectors *F1 ... Fn*.

The medium in the neighboring flow networks is defined over the *NetParam* component.

The minimum connection is:

- Emulsion/suspension inflow (*Nn*)
- Filter cake discharge (mixture fraction with low density or held back in the centrifuge chamber) (*Nn*)
- Filtrate discharge (mixture fraction expelled from the centrifuge chamber) (*Fn*)

Possible additional connections include:

- Inflow/discharge of cleaning agents
- Inflow/discharge of detergents
- Gas inflow and outflow
- Venting

For the water/steam connector type, the incoming medium is mixed internally in the tank with a liquid medium. The inflowing mass and enthalpy is taken into account but the model does not form a corresponding phase equilibrium. Incoming steam is added to the mass balance of the liquid phase and does not contribute to pressure build-up in the gas phase. Water is handled in the mixture like a liquid medium.

If water/steam is used as a discharge medium, it exits the centrifuge at the output with the mixture enthalpy of the liquid phase. All other calculations within this flow network are made using the properties of water/steam (e.g. heat capacity).

Due to the lack of steam properties, steam sterilization, for example, is currently not possible even at shutdown.

Parameter ConnectorHeight

For each of the connection points *N1..Nn* and *F1..Fn* you need the installation height of the nozzle on the centrifuge (parameter *ConnectorHeight* for connectors *N1..Nn* or *ConnectorHeightF* for connectors *F1..Fn*). The information is entered in meters. The default is 0 m and thus at the bottom of the centrifuge. Negative values are permitted for simulation of connectors at a lower position.

Input vector ScreeningRate

You can use the input vector *ScreeningRate* to specify the fraction of the non-separable mixture component of the inflowing medium, in percent, for each of the connectors *N1..Nn*. This will define the remaining fraction after separation for the emulsion or suspension inflow. Connectors without separation fractions (e.g. detergents) are given the default value "0". Gas connectors are also given the value "0".

The values defined here only apply to incoming substance flows. All outgoing substance flows exit with the non-separable mixture fraction of the total mass *Rate* currently stored in the centrifuge. This fraction is continuously calculated from the incoming and outgoing mass flows and the respective *ScreeningRate*.

In the case of outgoing mass flows, any *ScreeningRate* that may have been entered is not evaluated.

The filtrate outflows *F1..Fn* are always pure filtrate (i.e. *Rate* = 0.0). This means there is no input vector *ScreeningRate* for filtrate connector *F1..Fn*.

The input vector *ScreeningRate* is not visible by default.

Alternatively, you can also manually set the non-separable mixture component *Rate* of the **filled centrifuge** after the filling operation via the operating window.

Balance calculation of the liquid phase

All defined inputs in the connection vectors *N1 .. Nn* and *F1..Fn* must also be connected. None of these connectors may remain open during simulation because this will create open flow networks. This will result in an error message during code generation.

The liquid phase is balanced with respect to the inflowing and outflowing masses (m_{Liquid}) and enthalpies (h_{Liquid}). In addition, a mass balance of the non-separable mixture component ($m_{Screening}$) is carried out.

$$\frac{dm_{Liquid}}{dt} = \sum_{i=1}^{n} \dot{m}_{Liquid,i}$$

$$\frac{dm_{Screening}}{dt} = \sum_{i}^{n} \dot{m}_{Screening,i}$$

$$\frac{dh_{Liquid}}{dt} = \frac{1}{m_{Liquid}} \left(\sum_{i}^{n} \dot{m}_{Liquid,i}\, h_{Liquid,i} + \sum \dot{Q} \right)$$

$$T_{Liquid} = \frac{h_{Liquid}}{c_P}$$

Gas balance calculation

If at least one gas connector is connected, a corresponding gas pressure is also calculated. In this case, a closed and thus pressure-tight centrifuge is assumed. The gas pressure is changed by adding or removing gas or by changing the gas volume, e.g. by filling the tank. The calculated gas pressure also affects the liquid connectors.

The gas balancing applies exclusively to externally added (inert) gases. Vapor pressures from solvents, for example, are not taken into consideration.

The incoming and outgoing mass flows (m_{Gas}) and enthalpies (h_{Gas}) are balanced within the vapor space:

$$\frac{dm_{Gas}}{dt} = \sum_{i}^{n} \dot{m}_{Gas,i}$$

$$\frac{dh_{Gas}}{dt} = \frac{1}{m_{Gas}} \left(\sum_{i}^{n} \dot{m}_{Gas,i}\, h_{Gas,i} + \sum \dot{Q} \right)$$

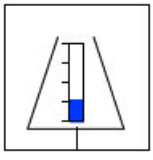
The gas pressure is then calculated according to the state equation of ideal gases:

$$p_{Gas} = \frac{m_{Gas}\, R_S\, T_{Gas}}{V_{Gas}}$$

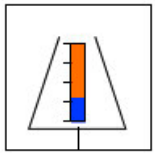
If a gas connector is not connected, the gas balance calculation does not take place. In this case, the gas pressure is replaced with the atmospheric pressure.

The component type *Centrifuge* has different symbols for operation with and without gas pressure.

Centrifuge without gas pressure



Centrifuge with gas pressure



A distinction is not made between suspension and emulsion within the component type. In case of a suspension, the internal calculation still takes place with two liquids. The corresponding removal of the filter cake of a suspension is thus also as a "liquid medium". Devices used for output (scrapers, screw conveyors, or similar) must therefore be replaced with valves and pumps in the model.

Operating modes of the centrifuge

The centrifuge is designed for continuous as well as batch operation.

Operation of the centrifuge is started with a corresponding standardized angular velocity (0% ... 100%) at the *Omega* input. Filtrate can only exit the centrifuge when the corresponding filtrate drain is open. The increase in angular velocity results in an increase of the filtrate discharge pressure. The discharge pressure is calculated in a simplified way from the radius of the centrifuge (based on geometric data) and the angular velocity.

Filtrate can still exit from a stopped centrifuge because no distinction is made between filtrate and, for example, cleaner. If this is to be prevented, an additional valve, for example, can be added to the model whose position is changed depending on the speed.

The absolute value of the filtrate mass flow can be set by means of k_V values in the filtrate drain (e.g. valve or nozzle).

Overwriting state variables

By using the operating window of the component, the operator can set the following state variables to any (permissible) value during simulation:

- Stored mass / level
- Temperature
- Non-separable mass fraction *Rate*

For this, the operator must first enter the setpoint in the corresponding digital input *SetValues* and apply it with the "Set" button. The state of the total stored mass can be specified using the value *MassTotal* as well as using the level (*Level*).

By manipulating state variables in this way, you can, for example, speed up heating processes or filling processes. As the operator, you can also empty the centrifuge, for example, to return to an earlier step in a step sequence.

Parameters

Parameter name	Description	Unit	Default value
Volume	Filling volume of the centrifuge	m^3	1.0
Height	Height of the filling volume of the centrifuge	m	1.0
NbrOfInletsOutlets	Number of general flow network connectors	–	2
NbrOfFiltrateOutlets	Number of filtrate connectors	–	1
NbrOfMeasurements	Number of measuring point connectors	–	1
MeasurementHeight [Nr]	Installation height of the measuring point connectors	m	0.0
ConnectorHeight [Nr]	Installation height of the general flow network connectors	m	0.0
ConnectorHeightF [Nr]	Installation height of the filtrate connectors	m	0.0

Additional parameters

Parameter name	Description	Unit	Default value
PressureOutside	Ambient pressure	bar	1.0
LevelInit	Initialization value for level of centrifuge	%	3.0
TemperatureInit	Initialization value for suspension temperature	°C	20.0
PressureInit	Initialization value for pressure in the case of a gas pressure centrifuge	bar	1.0
InitScreeningRateTotal	Initialization value for solids fraction in suspension (non-separable component)	%	0.0
Density	Density of the suspension	kg/m³	997.337
RemainderGas	Residual volume that cannot be filled with liquid when filling the centrifuge	m³	0.05
RemainderLiquid	Residual volume that remains in the pressure strainer after the strainer is emptied	m³	0.005
CpInit	Initial heat capacity for the medium stored in the centrifuge during initialization	–	4.18
geoHeight	Geodetic height; can be changed online	m	0.0

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit	Can variable be changed?
Length	L	m	Yes[1]
Temperature	T	°C	Yes[1]
Weight	M	kg	Yes[1]
Pressure	p_B	bar	No
Gas pressure	p_G	bar	No
Fraction of solids in solid/liquid mixture	w	%	Yes[1]

[1] Enable input with "Set"

Connection example

The figure below shows an example of the connection of a centrifuge:

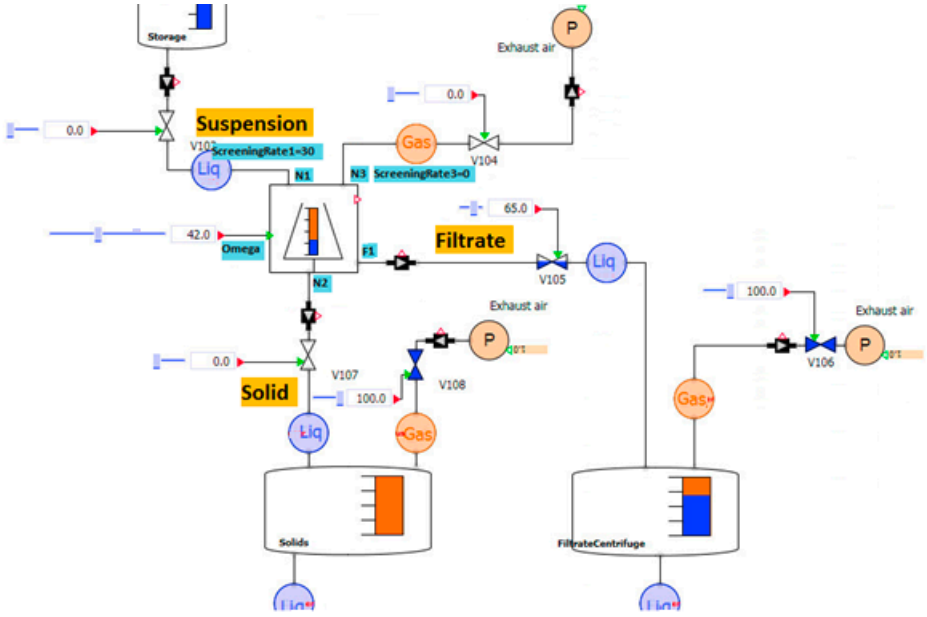


Figure 8-66 Connection example of component type Centrifuge

Make the following parameter settings for this example:

Centrifuge

	Name		Value
General	Volume	[m³]	1
Input	Height	[m]	1
Output	NbrOfInletsOutlets		3
Parameter	NbrOfFiltrateOutlets		1
Additional parameter	NbrOfMeasurements		1
State	▼ MeasurementHeight [1]		...
	MeasurementHeight1	[m]	0
	▼ ConnectorHeight [3]		...
	ConnectorHeight1	[m]	1
	ConnectorHeight2	[m]	-2
	ConnectorHeight3	[m]	1
	▼ ConnectorHeightF [1]		...
	ConnectorHeightF1	[m]	0

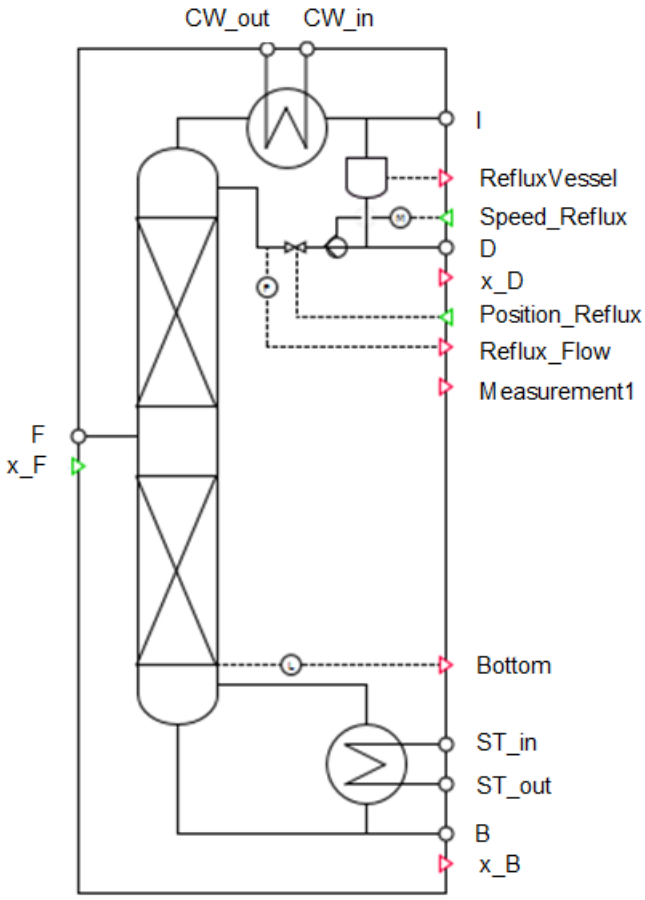
Figure 8-67 Parameter settings of component type Centrifuge in the connection example

See also

Geodetic height (Page 618)

Column

Symbol



Symbol of component type Column

Function

The *Column* component type calculates the distillation of a binary feed to a low boiling mixture (top product) and a high boiling mixture (bottom product) in a plate column.

The feed flow enters the column through the *F* connection with the concentration of low-boiler *x_F*. The top product (distillate) is taken from the column at connection *D* with the concentration of low-boilers *x_D*. The bottom product (bottom) is taken from the column at connection *B* with the concentration of low-boilers *x_B*.

There is a condenser with a reflux vessel at the top of the column. The condenser has two connections for cooling water (*CW_in* and *CW_out*). A pump (either in the outlet of the reflux vessel or in the reflux pipe) pumps the reflux into the column. Non-condensable parts of the top product are removed from the column through connection *I*.

In the sump, the column is heated by a natural convection evaporator (reboiler). The reboiler has two connections for heating steam (*ST_in* and *ST_out*).

The separation of low boilers and high boilers is calculated idealized using so-called split factors. The split factors are determined using empirical formulas depending on reflux, boilup, feed, and feed concentration compared to the (stationary) operation point (OP). The stationary operation point is specified with the following parameters:

- Feed flow: *OP_mflF*
- Feed concentration: *OP_xF*
- Top concentration: *OP_xD*
- Bottom concentration: *OP_xB*
- Reflux: *OP_mflR*
- Boilup: *OP_mflV*
- Evaporator capacity: *OP_Qreb*

The split factors are limited with the *Split_Max* parameter. The maximum split factor is *Split_Max* and the minimum is (1 – *Split_Max*). The greater the individual split factor, the more low boilers or high boilers are separated over the top product.

The temperatures in the column are calculated depending on concentration of low boiler x and the boiling point of low and high boiler (*TS_LB* and *TS_HB*) at operation pressure.

$$T = \left(x T_{S_{LB}}\right) + \left((1-x) T_{S_{HB}}\right)$$

The enthalpy of evaporation *hV* is calculated as a function of the concentration of low boiler *x* and the enthalpy of evaporation of the low and high boiler (*hV_LB* and *hV_LB*).

$$h_V = \left(x h_{V_{LB}}\right) + \left((1-x) h_{V_{HB}}\right)$$

The natural convection in the reboiler is calculated on the basis of the evaporator capacity *Q_Reb* compared to the reflux amount *OP_mflV* and evaporator capacity *OP_Qreb* at the stationary operating point. The evaporator capacity is calculated on the basis of the heat transferred by the heating medium in the reboiler.

The column size is specified with the following parameters:

- Diameter of the column: *Diameter*
- Number of trays: *n_tray*
- Tray height: *Tray_Height*

The feed tray of the column is specified with the *Feed_Tray* parameter.

The parameter *Pump_Top* can be used to select whether the pump is only in the reflux pipe (*Reflux*) or if it is located in the outlet of the reflux tank for reflux and distillate (*Reflux +Distillate*).

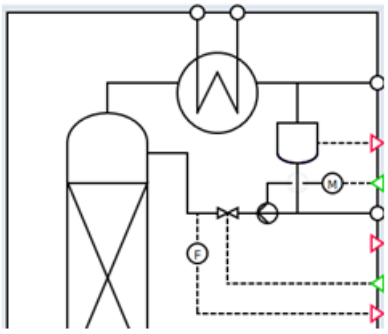
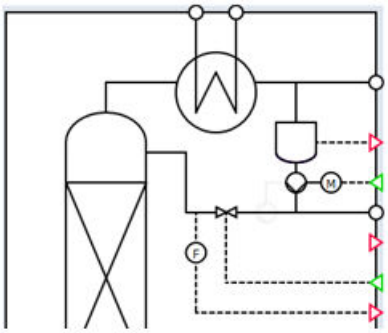


Figure: Pump in reflux pipe



Pump for reflux and distillate

The top condenser is parameterized with the *A_Cond* exchange area and *k_Cond* heat transfer coefficient parameters.

The reflux vessel is parameterized using the parameters *Diameter_RefluxVessel* and *Volume_RefluxVessel*. If the reflux vessel is running empty, it is displayed with blue lines. If the vessel is full and overflows, it is displayed with red lines.

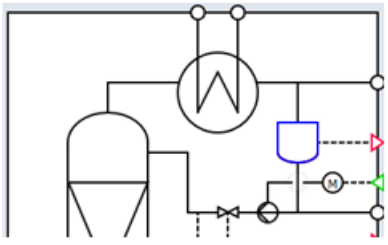
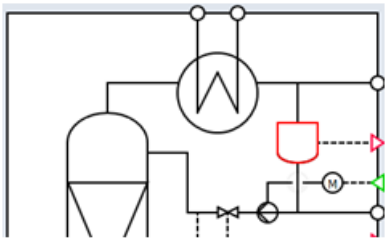


Figure: Reflux vessel empty



Reflux vessel full

The reboiler is configured with the *A_Reb* exchange area and *k_Reb* heat transfer coefficient parameters.

The level, pressure and temperature in the reflux vessel can be monitored via the measuring point connection *RefluxVessel*.

The flow of the reflux can be monitored via the measuring point connection *Reflux_Flow*.

The level, pressure and temperature in the bottom of the column can be monitored via the measuring point connection *Bottom*.

Measurements (pressure and temperature) at the column can be monitored via the measuring point connectors *Measurement[i]*. The number can be specified with the *NbrOfMeasurements* parameter. The tray with the measuring point can be specified for each measuring point with the *MeasurementTray* parameter.

The pump speed can be specified with the *Speed_Reflux* input. The speed can be between 0 (pump stopped) and 100 (full speed).

The valve position of the valve in the reflux pipe can be specified with the *Position_Reflux* input. The valve position can be specified between 0 (valve closed completely) and 100 (valve opened completely).

Parameters

Parameter name	Description	Unit	Default value
Diameter	Column diameter	m	2
N_tray	Number of trays	–	10
Tray_Height	Distance between trays	mm	50
Feed_tray	Feed tray	–	10
A_Cond	Top condenser exchange area	m^2	50
K_Cond	Top condenser heat transfer coefficient	$W/m^2/K$	1000
ZeroFlowHead_Reflux	Pump zero flow head at reflux vessel	bar	5
NominalPressure_Reflux	Nominal pressure of the pump at the reflux vessel	bar	4
NominalMassflow_Reflux	Nominal mass flow of the pump at the reflux vessel	kg/s	5
Diamter_RefluxVessel	Reflux vessel diameter	m	1
Volume_RefluxVessel	Reflux vessel volume	m^3	1
geoHeight_RefluxVessel	Geodetic height, reflux vessel	m	0
Pump_Top	Pump configuration at reflux vessel	Reflux flow	
OP_mflF	Feed mass flow at operating point	kg/s	6
OP_xF	Feed concentration at operating point	kg/kg	0.4
OP_xD	Distillate concentration at operating point	kg/kg	0.9
OP_xB	Bottom product concentration at operating point	kg/kg	0.1
OP_mflR	Reflux mass flow at operating point	kg/s	4.5
OP_mflV	Boilup mass flow at operating point	kg/s	17
OP_QReb	Reboiler heating capacity at operating point	kW	7000
A_Reb	Reboiler exchange area	m^2	400
K_Reb	Reboiler heat transfer coefficient	$W/m^2/K$	1000
geoHeight_Colum	Column geodetic height (at the sump)	m	0
NbrOfMeasuremnsts	Number of measurement points of the column	–	1
MeasurementTray	Tray on which the measuring point is located	–	10

Additional parameters

Parameter name	Description	Unit	Default value
LiqLevTray	Liquid level at column tray	mm	50
hV_LB	Low-boiler enthalpy of evaporation	kJ/kg/K	1100
hV_HB	High-boiler enthalpy of evaporation	kJ/kg/K	2200
TS_LB	Low-boiler boiling point at column pressure	°C	65

Parameter name	Description	Unit	Default value
TS_HB	High-boiler boiling point at column pressure	°C	100
DTmin	Minimum temperature difference, heat exchanger	K	5
Split_Max	Maximum separation factor	–	0.9999
LevelPercent_RefluxVessel_Init	Reflux vessel level initialization	%	50
LevelPercent_Bottom_Init	Initialization level column bottom	%	50
kvs_CW	Resistance coefficient, top condenser flow, service medium	m^3/h	200
kvs_Reflux	Resistance coefficient, reflux valve	m^3/h	10
pInit_Top	Initialization pressure at top of the column	bar	1
PInit_Bottom	Initialization pressure at bottom of the column	bar	1.1

Operating window

The main parameters of the column are displayed in the operating window. The following variables are displayed:

Variable	Symbol	Unit
Top pressure	p_Kopf	bar
Top temperature	T_Kopf	°C
Top condenser cooling capacity	Q_Cond	kW
Reflux mass flow	mfl_R	kg/s
Gas mass flow top	mfl_I	kg/s
Top product flow	mfl_D	kg/s
Top product concentration	x_D	kg/kg
Reflux ratio	RR	–
Feed mass flow	mfl_F	kg/s
Feed concentration	x_F	kg/kg
Boilup ratio	BR	–
Boilup mass flow	mfl_V	kg/s
Evaporator capacity	Q_Reb	kW
Bottom temperature	T_Sumpf	°C
Bottom pressure	p_sumpf	bar
Bottom product flow	mfl_B	kg/s
Bottom product concentration	x_B	kg/kg

Column Tower

Symbol

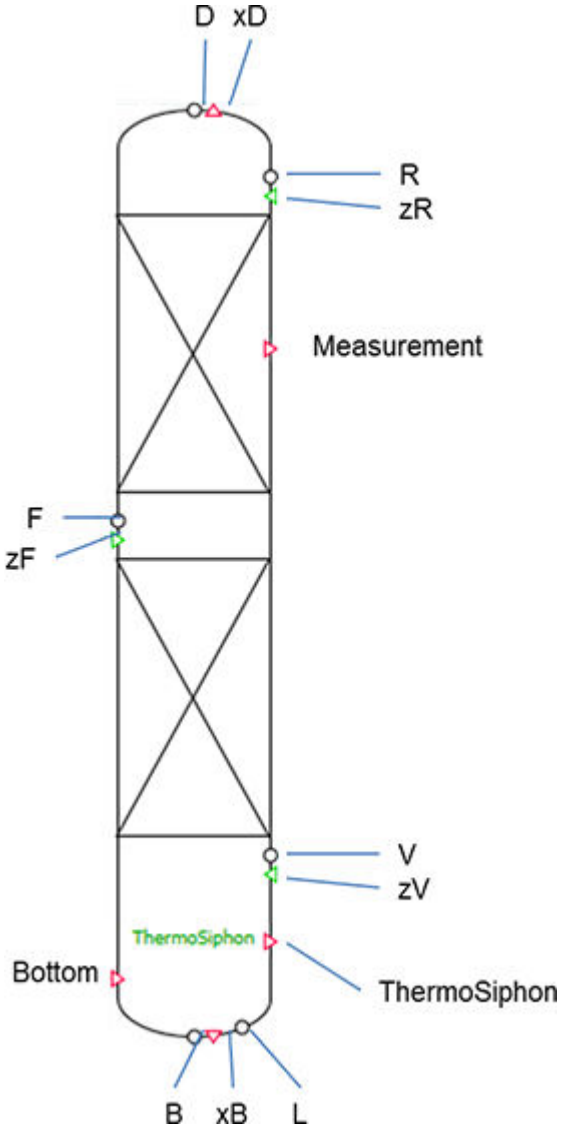


Figure 8-68 Symbol of component type Column Tower

Function

The *Column Tower* component type calculates the distillation of a binary feed to a low boiling mixture (top product) and a high boiling mixture (bottom product) in a plate column.

The feed flow is supplied to the column over the *F* connector with the concentration of low boilers *zF*. The top product (distillate) is taken from the column at the *D* connector with the concentration of low boilers *xD*. The bottom product (bottom) is taken from the column at the *B* connector with the concentration of low boilers *xB*. The reflux flow is supplied at the *R* connector with the concentration *zR*. The bottom flow to the reboiler is removed at the *L* connector. The

heated and partially evaporated flow from the reboiler is once again returned to the column at the *V* connector. The concentration of low boilers for this flow is input via the *zV* connector.

The separation of low boilers and high boilers is calculated idealized using so-called split factors. The split factors are determined using empirical formulas depending on reflux, boilup, feed, and feed concentration compared to the (stationary) operation point (OP). The stationary operation point is specified with the following parameters:

- Feed *OP_mflF*
- Feed concentration *OP_xF*
- Top concentration *OP_xD*
- Bottom concentration *OP_xB*
- Reflux flow *OP_mflR*
- k_{VS} value evaporator *OP_kvReb* (for natural circulation) or
- Circulation flow *OP_mflCircForced* (for forced circulation)

The split factors are limited with the *Split_Max* parameter. The maximum split factor is *Split_Max*, the minimum split factor is (1 – *Split_Max*). The greater the individual split factor, the more low boilers or high boilers are separated over the top product.

The temperatures in the column are calculated based on the concentration of the low boiler *x* and the boiling points of the low boiler and high boiler (*TS_LB* and *TS_HB*) as well as the spread *S* of the boiling lens at column operation pressure.

$T = f\,(TS_{LB}, TS_{HB}, S, x)$

The *ThermoSiph* connector of the column can be connected to the *ThermoSiph* connector of the evaporator/reboiler. The necessary information such as vapor content *vf* and composition of the vapor and liquid phase (*x* and *y*) at the reboiler output as well as the bottom concentration of the *xB* column is then transferred directly. If this connector is connected, you no longer need to specify the input *zV*.

The *ThermoSiphon* parameter can be used to set that the reboiler is operated as a thermosiphon reboiler. In this case the *ThermoSiph* connector of the column can be connected to the *ThermoSiph* connector of the evaporator. The natural circulation between column and evaporator is then implemented accordingly. If the *ThermoSiphon* parameter is not set, we assume a forced circulation with pump. The *ThermoSiph* connector can also be connected to the evaporator here.

The column size is specified with the following parameters:

- Column diameter *Diameter*
- Number of trays *n_tray*
- Tray height *Tray_Height*

The feed tray of the column is specified with the *Feed_Tray* parameter.

The reflux vessel is configured with the diameter *Diameter_RefluxVessel* and volume *Volume_RefluxVessel* parameters. If the reflux vessel runs empty, the vessel is shown with blue lines. If the vessel is completely filled and overflows, the vessel is shown with red lines (see figure).

The level, pressure and temperature in the bottom of the column can be monitored via the measuring point connection *Bottom*.

Measurements (pressure and temperature) at the column can be read out over the measuring point connectors *Measurement[i]*. The number can be specified with the *NbrOfMeasurements* parameter. The tray on which the measuring point is located can be specified for each measuring point with the *MeasurementsTray* parameter.

Parameter

The following parameters are available:

Parameter name	Description	Unit	Default value
Diameter	Column diameter	m	2
n_Tray	Number of trays	–	10
Tray_Height	Distance between trays	mm	50
Feed_tray	Feed tray	–	10
NbrOfMeasuremnsts	Number of measuring points on the column	–	1
MeasurementTray	Tray on which the measuring point is located	–	10
OP_mflF	Feed mass flow at operating point	kg/s	6
OP_xF	Feed concentration at operating point	kg/kg	0.4
OP_xD	Distillate concentration at operating point	kg/kg	0.9
OP_xB	Bottom product concentration at operating point	kg/kg	0.1
OP_mflR	Reflux mass flow at operating point	kg/s	4.5
geoHeight_Colum	Geodetic height of the column (at the bottom)	m	0
H_Reb	Length of the reboiler	mm	2000
ThermoSiphon	Switch for natural circulation	–	True
OP_TF	Feed temperature at the operating point	°C	60
OP_TR	Reflux flow temperature at the operating point	°C	60
OP_xR	Reflux flow concentration at the operating point	kg/kg	0.9
OP_mflCircForced	Mass flow forced circulation at the operating point	kg/s	20

Additional parameters

The following additional parameters are available:

Parameter name	Description	Unit	Default value
T_LB	Low-boiler boiling point at column pressure	°C	65
T_HB	High-boiler boiling point at column pressure	°C	100
hV_LB	Low-boiler enthalpy of evaporation	kJ/kg/K	1100
hV_HB	High-boiler enthalpy of evaporation	kJ/kg/K	2200
S	Spread of boiling lens	–	10
pInit_Top	Initialization pressure at top of the column	bar	1
pInit_Bottom	Initialization pressure at bottom of the column	bar	1.1
LiqLevTray	Liquid level at column tray	mm	50
Split_Max	Maximum separation factor	–	0.9999
InitLevelBottom	Initialization level column bottom	%	50
kvsLiq	Resistance coefficient liquid connectors	m^3/h	1000
kvsVap	Resistance coefficient gas connectors	m^3/h	100000
kv0	Resistance coefficient connectors opposite intended flow direction (check valve)	m^3/h	0.000001

Operating window

The main parameters of the column are displayed in the operating window. The following variables are displayed:

Variable	Symbol	Unit
Top pressure	pKopf	bar
Top temperature	TKopf	°C
Reflux mass flow	mflR	kg/s
Reflux flow concentration	xR	kg/kg
Top product flow	mflD	kg/s
Top product concentration	xD	kg/kg
Reflux ratio	RR	–
Feed mass flow	mflF	kg/s
Feed concentration	xF	kg/kg
Boilup ratio	BR	–

Variable	Symbol	Unit
Boilup mass flow	mflV	kg/s
Boilup concentration	xV	kg/kg
Bottom temperature	T_Sumpf	°C
Bottom pressure	p_sumpf	bar
Bottom product flow	mflB	kg/s
Bottom product concentration	xB	kg/kg
Pressure drop column	DeltaP	mbar

Column tower with side stream

Symbol

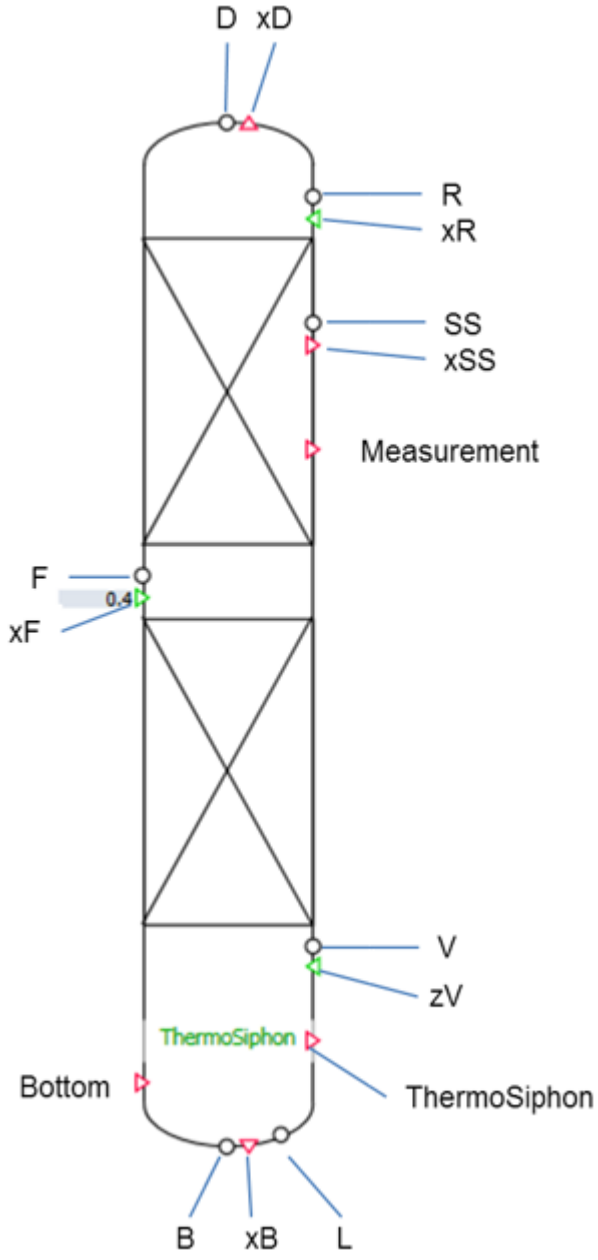


Figure 8-69 ColumnTowerwithSideStream

Function

The *Column Tower with Side Stream* component type uses a simplified calculation to calculate the distillation of a binary feed into a low boiling mixture (top product) and a high boiling mixture (bottom product) in a plate column. In addition, a side stream is removed.

The feed flow enters the column via connector *F* with low-boiler concentration *zF*. The top product (distillate) is removed from the column at connector *D* with low-boiler concentration *zD*. The bottom product *(Bottom)* is removed from the column at connector *B* with low-boiler concentration *zB*. The reflux is fed at connector *R* with the concentration *zR*. The bottom flow to the reboiler is removed at connector *L*. The heated and partially evaporated flow from the reboiler is fed back to the column at connector *V*. The low-boiler concentration for this flow is input via connector *zV*. The side stream is removed from the column at connector *SS* with light-boiler concentration *xSS*. The removed side stream is always liquid.

The separation of low boilers and high boilers is calculated idealized using so-called split factors. The split factors are determined using empirical formulas depending on reflux, boilup, feed, and feed concentration compared to the (stationary) operation point (OP). The stationary operating point of the column is specified with the following parameters:

- Feed *OP_mflF*
- Feed concentration *OP_xF*
- Top concentration: *OP_xD*
- Bottom concentration: *OP_xB*
- Reflux *OP_mflR*
- Kvs value evaporator *OP_kvReb* (for natural circulation)
- Circulation flow *OP_mflCircForced* (for forced circulation)
- Side stream *OP_mflSS*
- Side stream concentration *OP_xSS*

The split factors are limited with the *Split_Max* parameter. The maximum split factor is *Split_Max* and the minimum is (1 - *Split_Max*). The greater the individual split factor, the more will be separated from the low boilers or high boilers over the top product.

The temperatures in the column are calculated as a function of the concentration of the low boiler *x* and the low-boiler and high-boiler boiling points (*TS_LB* and *TS_HB*) as well as the spread *S* of the boiling curve at column operation pressure.

$$T = f(TS_{LB}, TS_{HB}, S, x)$$

Connector *ThermoSiph* of the column can be connected to connector *ThermoSiph* of the evaporator/reboiler. The necessary information such as vapor content *vf* and composition of the vapor and liquid phase (*x* and *y*) at the reboiler output as well as the bottom concentration of the *xB* column is then transferred directly. If this connector is connected, you no longer need to specify the input *zV*.

Operation of the reboiler as a natural circulation evaporator can be set using the *ThermoSiphon* parameter. In this case, connector *ThermoSiph* of the column must be connected to connector *ThermoSiph* of the evaporator. The natural circulation between column and evaporator is then implemented accordingly. If the *ThermoSiphon* parameter is not set, forced circulation with a pump is assumed. Connector *ThermoSiph* can also be connected to the evaporator here.

The column size is specified with the following parameters:

- Diameter of the column *Diameter*
- Number of trays *n_tray*
- Distance between trays *TrayHeight*

The feed tray of the column is specified with the *Feed_Tray* parameter. The extraction tray for the side stream is specified with the *TraySideStream* parameter. If the *TraySideStream* parameter is set to 0, the column is calculated without a side stream. Connector *SS* must be interconnected nevertheless because an open flow network is otherwise created.

The level, pressure and temperature in the column sump can be read out using the measuring point connection *Bottom*.

Measurements (pressure and temperature) in the column can be read out using measurement point connector *Measurement[i]*. The *NbrOfMeasurements* parameter can be used to specify the number. The tray on which the measurement point is located can be specified for each measurement point using the *MeasurementsTray* parameter.

Parameters

Parameter name	Description	Unit	Default value
Diameter	Column diameter	m	2
N_tray	Number of trays	-	10
Tray_Height	Distance between trays	mm	50
Feed_tray	Feed tray	-	10
NbrOfMeasurements	Number of measurement points of the column	-	1
MeasurementTray	Tray on which the measurement point is located	-	10
OP_mflF	Feed mass flow at operating point	kg/s	6
OP_xF	Feed concentration at operating point	kg/kg	0.4
OP_xD	Distillate concentration at operating point	kg/kg	0.9
OP_xB	Bottom product concentration at operating point	kg/kg	0.1
OP_mflR	Reflux mass flow at operating point	kg/s	4.5
H_Reb	Length of the reboiler	mm	2000
ThermoSiphon	Switch for natural circulation	-	True
OP_TF	Feed flow temperature at operating point	°C	60
OP_TR	Reflux flow temperature at operating point	°C	60
OP_xR	Concentration in reflux flow at operating point	kg/kg	0.9
OP_mflCircForced	Forced circulation flow at operating point	kg/s	20
TraySideStream	Tray of side stream	-	0

OP_mflSS	Mass flow of side stream at operating point	kg/s	0
OP_xSS	Low-boiler concentration of side stream at operating point	kg/kg	0

Additional parameters

Parameter name	Description	Unit	Default value
T_LB	Low-boiler boiling point at column pressure	°C	60
T_HB	High-boiler boiling point at column pressure	°C	100
hV_LB	Low-boiler enthalpy of evaporation	kJ/kg/K	1800
hV_HB	High-boiler enthalpy of evaporation	kJ/kg/K	2300
S	Spread of the boiling curve	-	10
pInit_Top	Initialization pressure at top of the column	bar	1
PInit_Bottom	Initialization pressure at bottom of the column	bar	1.1
LiqLevTray	Liquid level at column tray	mm	50
Split_Max	Maximum split factor	-	0.9999
InitLevelBottom	Initialization level of column sump	%	50
KK	Proportionality factor for natural circulation	-	3
Ti	Time constant for natural circulation	-	100
kvsLiq	Resistance coefficient for liquid connectors	m^3/h	1000
kvsVap	Resistance coefficient for gas connectors	m^3/h	100000
Kv0	Resistance coefficient for connectors opposite intended flow direction (check valve)	m^3/h	1e-6
geoHeight	Geodetic height of the column (at the sump)	m	0

Operating window

The main parameters of the column are displayed in the operating window.

Variable	Symbol	Unit
Top pressure	pKopf	bar
Top temperature	TKopf	°C
Reflux mass flow	mflR	kg/s
Concentration in reflux flow	xR	kg/kg

Top product flow	mflD	kg/s
Top product concentration	xD	kg/kg
Reflux ratio	RR	-
Feed mass flow	mflF	kg/s
Feed concentration	xF	kg/kg
Boil-up flow ratio	BR	-
Boil-up mass flow	mflV	kg/s
Concentration in boil-up	xV	kg/kg
Sump temperature	T_Sumpf	°C
Sump pressure	p_sumpf	bar
Bottom product flow	mflB	kg/s
Concentration in bottom product	xB	kg/kg
Side stream	mflSS	kg/s
Concentration in side stream	xSS	kg/kg

Crystallizer

Symbol

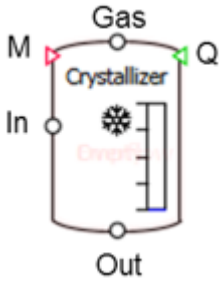


Figure 8-70 Crystallizer

Function

The component type *Crystallizer* calculates the crystallization of a substance from a mass flow. This can take place through evaporation of the solvent (evaporation crystallization) as well as through cooling of the mixture (cooling crystallization). The incoming and outgoing mass flows are balanced with respect to mass and enthalpy. In addition, a mass balance of the dissolved crystals in the solvent and the crystal itself is performed.

A direct input can be used for temperature control of or add heat to the accumulated mixture in the crystallizer. The mixture, consisting of solvent and dissolved crystals, is fed to the *Crystallizer* via connector *In*. The mixture, consisting of solvent and dissolved crystals and crystals, exits the crystallizer via connector *Out*. A flow network with gas medium must not be connected to connectors *In* and *Out*.

The pressure in the crystallizer is applied and any evaporating solvent is removed via connector *Gas*. A flow network with the gas or water/steam medium must be connected to this connector.

In the case of water/steam, the process conditions in the *Crystallizer* must be such that the water is in vapor form.

The medium in the adjoining flow networks is defined using the *NetParam* component. All connections must be interconnected. None of these connectors may remain open during simulation because this would create open flow networks. This would, in turn, result in an error message during code generation.

Evaporation

To evaporate solvent, the mixture must be heated to a temperature that is above the boiling point corresponding to the current pressure. The vapor pressure curve needed for this is configured using the *VaporPressure* parameter (vapor pressure curve of low-boiler $p_S = f\ (T_S)$ in bar and °C). In addition, the enthalpy of vaporization of the solvent must be specified as a characteristic curve ($\Delta H_V = f(T_S)$ in kJ/kg and °C) using the *EnthalpyOfVaporization* parameter.

These characteristic curves are preset for water. They can be edited within the charts or read in from a text file. Values on the abscissa are not extrapolated. Examples of these characteristic curves are shown in the figures below:

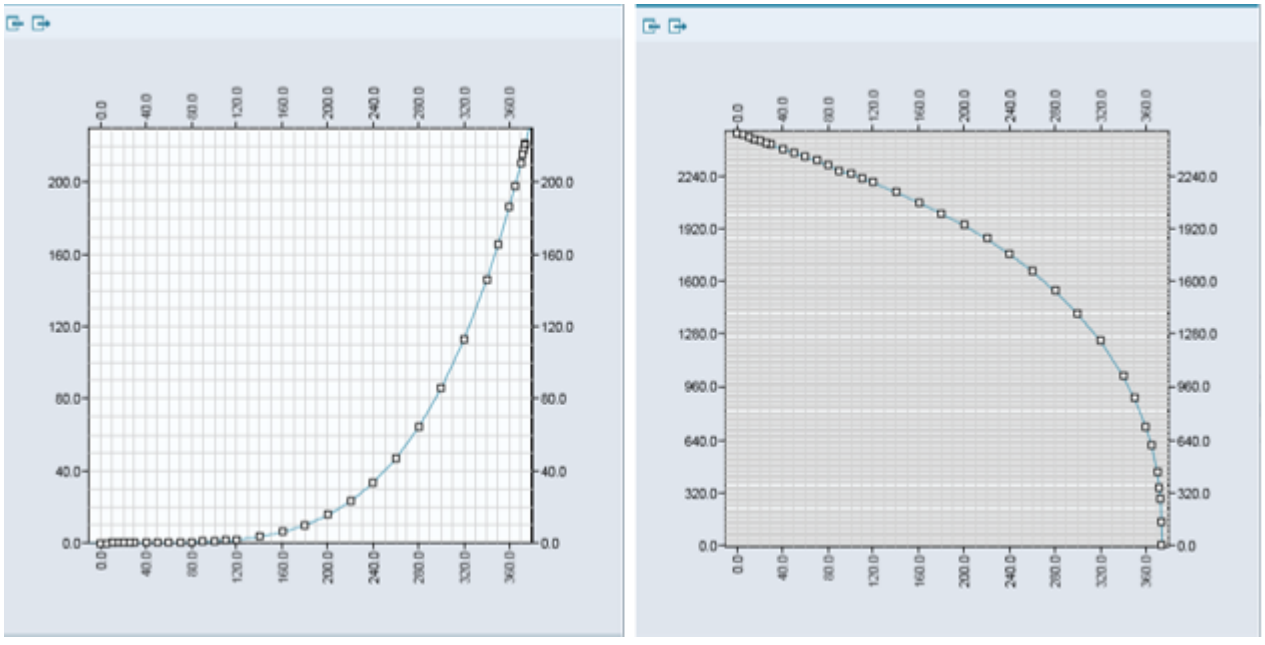


Figure 8-71 Example of enthalpy of vaporization and vapor pressure curve for crystallizer

When the evaporation condition $T > T_S$ (p_S) is reached, the solvent gradually changes to the gas phase and is accounted for as gas in the mass balance. No provision is made for condensation in the tank.

The evaporation quantity $\dot{m}_{Evap}$ is calculated as a function of the pressure difference between vapor pressure and pressure in the crystallizer and the temperature T:

$$\dot{m}_{Evap} = f_{Evap} \cdot \frac{\Delta P}{R_m \cdot T} \cdot A$$

f_{Evap}	Evaporation factor *EvapFactor*
R_m	Specific gas constant
A	Cross-sectional area of the tank

The *EvapRateVariability* parameter can be used to adapt the rate of change of the evaporation quantity. If values are too low, the state of equilibrium may be reached only very slowly or may not be reached at all. If values are too high, instability may result.

Crystallization

In order for crystals to crystalize from the mixture, the maximum solubility of the crystals in the solvent must be exceeded. The solubility curve needed for this is configured with the *Solubility* parameter (solubility of crystals in the solvent in *kg*/*kg* and °C). The characteristic curve for the solubility of NaCl in water is preset. This characteristic curves is shown in the figure below as an example:

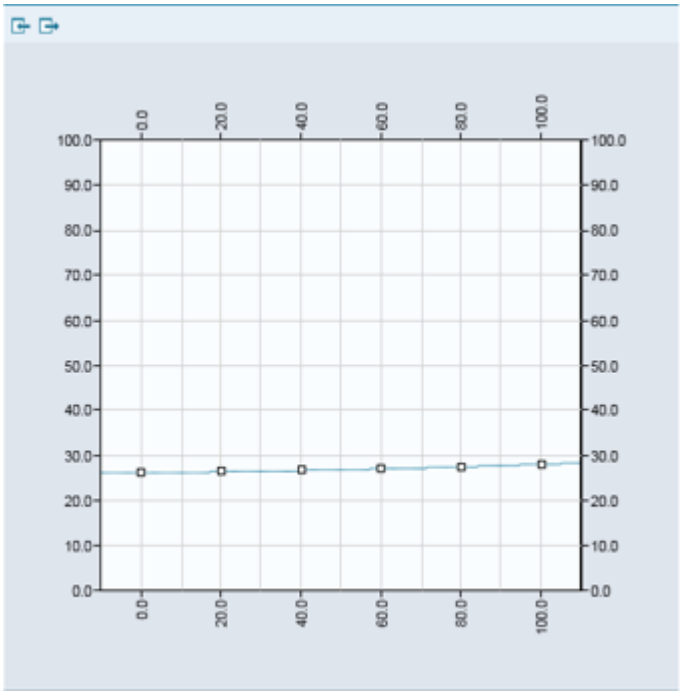


Figure 8-72 Example of the solubility curve:

When the mass concentration of dissolved crystal $w_{DSC} > w_S$, the supersaturated quantity of crystal precipitates directly. A crystal growth rate or a metastable range was not taken into account.

The specification of the mass concentration w_{DSC} relates to the quantity of dissolved crystal (DSC) in the solvent:

$$w_{DSC} = \frac{m_{DSC}}{m_L}$$

w_{DSC} Mass concentration of dissolved crystal
m_{DSC} Mass of dissolved crystal
m_L Mass of solvent (including dissolved crystal)

The specification of the mass concentration w_C relates to the quantity of precipitated crystal of the total mass:

$$w_C = \frac{m_C}{m_L + m_C} = \frac{m_C}{m_{tot}}$$

w_C Mass concentration of precipitated crystal
m_C Mass of precipitated crystal
m_{tot} Total mass

The *HeatOfCrystallization* parameter can be used to specify the heat released by the crystallization.

Borderline case "empty crystallizer"

Inflows into the crystallizer and outflows from the crystallizer are generally throttled at the nozzle of the crystallizer. During normal operation, this throttling is not immediately recognizable for the user, because throttling should have very little impact on the simulation. For an empty crystallizer, the outflow becomes strongly throttled. The flow coefficients are internally calculated as a function of the crystallizer volume and cannot be influenced by the modeler. The crystallizer is considered empty when the liquid volume in it is less than a specified minimum crystallizer filling in percent *MinLiquidVolumePercent*.

The balancing of the states is limited for an empty crystallizer. That is, changes of the accumulated mass are discarded. No evaporation takes place and the energy supply via connector *Q* is neglected. The "empty" state continues until there is a sufficient increase in the liquid quantity. That is, the result of inflowing quantity minus outflowing quantity is positive. A corresponding indication of an empty crystallizer is shown in the component symbol.

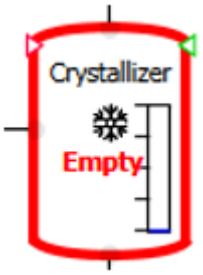


Figure 8-73 Indication of an empty crystallizer

Borderline case "full crystallizer"

The crystallizer is considered to be full when its liquid volume reaches the maximum possible percentage value, so that the gas volume is less than a given minimum crystallizer filling in percent *MinGasVolumePercent*.

The balancing of the states is limited for a full crystallizer. That is, changes of the accumulated mass are discarded. No evaporation takes place and the energy supply via connector *Q* is neglected. The "full" state continues until there is a sufficient decrease in the liquid quantity. That is, the result of inflowing quantity minus outflowing quantity is negative. A corresponding indication of a full crystallizer is shown in the component symbol.

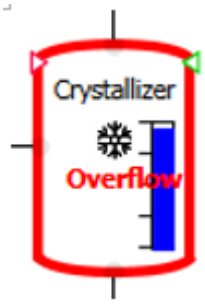


Figure 8-74 Indication of the full crystallizer

Parameters

Parameter name	Description	Unit	Default value
Volume	Volume V of the crystallizer	m^3	1.0
Height	Height of the crystallizer	m	1.0
VaporPressure	Vapor pressure curve of the solvent: Pressure in bar as a function of temperature (°C)	-	
EnthalpyOfVaporization	Enthalpy of vaporization of the solvent in kJ/kg as a function of temperature (°C)	-	
Solubility	Solubility curve of crystals in the solvent in kg/kg as a function of temperature (°C)	-	

Additional parameters

Parameter name	Description	Unit	Default value
HeatOfCrystallization	Heat of crystallization	kJ/kg	0
LevelInit	Initialization level	%	10.0
TemperatureInit	Initialization temperature	°C	20.0
PressureInit	Initialization pressure	bar	1
HeatCapacityInit	Initialization heat capacity	kJ/kgK	4.18

MoistureInit	Initialization mass fraction of moisture	%	0.0
BulkDensity	Density of heavy phase	kg/m³	1000.0
EvapFactor	Evaporation factor	-	1.0
EvapRateVariability	Variability of evaporation rate	-	1.0
MinGasVolumePercent	Percentage of crystallizer volume that cannot be filled with liquid when filling the crystallizer	%	5
MinLiquidVolumePercent	Percentage of crystallizer volume that remains in the crystallizer when the crystallizer is emptied	%	0.5
geoHeight	Geodetic height of the crystallizer	m	0.0

Operating window

Variable	Symbol	Unit
Level	L	m
Total mass	M_{tot}	kg
Temperature of the liquid	T_L	°C
Gas pressure	P_G	bar
Mass fraction of dissolved crystals in the solvent	w_{DSC}	%
Mass fraction of crystals	w_C	%
Mean residence time	t	s

Filter

Symbol

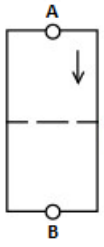


Figure 8-75 Symbol of component type Filter

Function

The *Filter* component simulates the behavior of a filter in a pipeline in the form of a pressure drop.

This pressure drop depends on:

- Flow coefficient of the clean filter (k_{VS} value)
- Throughflow (volume flow, medium)
- Clogging of the filter (*Clogging*)

The pressure drop across the filter is calculated according to the following formula:

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_{VS}^2\ (1-clogging)^2\ \rho\frac{\text{kg}}{\text{m}^3}}\ 12960\left(\frac{\text{sec}}{\text{h}}\right)^2$$

Whereby:

$\Delta p = p_B - p_A$	the pressure drop across the filter in bar
$\dot{m}$	the flow or mass flow in kg/s
ρ	the density of the medium in kg/m³
k_{VS}	the flow coefficient of the filter in m³/h
clogging	the degree of clogging of the filter [0..1]

For the mass flow $\dot{m}$, the reference direction is defined from connector *A* to connector *B*, which means for a mass flow in reference direction $\dot{m} > 0$. This means the pressure drop therefore has a negative value: $\Delta p < 0$.

This fact is also represented by an arrow in the symbol.

You can configure a clogging rate using the *CloggingRate* parameter. This parameter indicates the clogging rate in percentage per mass passing through. You can use this to model a filter whose clogging rate and therefore pressure drop increase at a certain rate.

To simplify matters, we assume that the incoming mass flow is equal to the outgoing mass flow at any time. The partial mass flow filtered out from the main flow is thus neither removed from the main mass flow nor added or taken away.

You can clean the filter using the "Clogging Reset" button. This function can also be connected externally using the *Reset* input (e.g. after expiration of a sequence for purging the filter).

Parameters

Parameter name	Description	Unit	Default value
Kvs	Flow coefficient of the clean filter k_{VS} with $k_{VS} \geq 10^{-6}$ m³/h	m³/h	10.0
CloggingRate	Change in degree of clogging per flow-through quantity (specified in kg) in %	%/kg	0.01

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure drop	Δp	bar
Mass flow	$\dot{m}$	kg/s
Clogging[1]	–	%

[1] Can be reset with "Reset"

PressureStrainer

Symbol

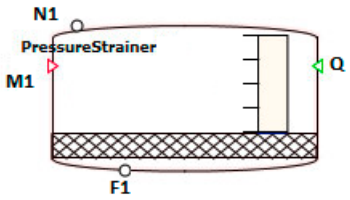


Figure 8-76 Symbol of component type PressureStrainer

Function

The *PressureStrainer* component type simulates the functionality of a pressure strainer.

The pressure strainer is a filter in which a suspension is mechanically separated. To do this, the pressure strainer is filled with a suspension. The pressure strainer is then subjected to an overpressure, which presses the filtrate through a filter, and thus achieves the separation.

Additional steps can be executed between filling the strainer and applying the pressure, such as

- Sedimentation of the solids on the bottom
- Smoothing of the layer formed on the bottom by sedimentation with the aid of an agitator

The pressure strainer component is modeled as ideally mixed. The above-named steps can thus not be simulated. However, corresponding actions can be simulated in the actuator/sensor level, such as:

- Wait times of the agitator
- Activities of the agitator (CCW / CW rotation, speeds, agitation time, etc.)

The effects of these actions are generally not directly measurable in the pressure strainer.

If you need individual effects from these actions, you must model them additionally using logic blocks.

By applying pressure to the strainer, the filtrate is slowly pressed through the filter. As soon as a configurable level of dryness *DryingRate* is reached in the filter cake, a "breakthrough" of the gas occurs. The gas thereby escapes from the strainer space through drying cracks in the filter cake.

The pressure drop associated with this process represents the end of filtration for the component. Additional drying is now only possible by heating the pressure strainer.

The filter cake is output as liquid medium. Devices used for output (scrapers, screw conveyors, or similar) must therefore be replaced with valves and pumps in the model.

The component type *PressureStrainer* offers the option of connecting a heating or cooling jacket for drying (*HeatingJacket*, *ElectricalHeatingJacket* component). A heat input, e.g. by means of an agitator, is possible using input *Q*.

Within the component type *PressureStrainer*, the separation occurs in a simplified manner by removal of the various fractions (filtrate and filter cake) from partial mass balances. The structure of the filter cake and the actual filter bottom are not modeled.

Circuit

The component *PressureStrainer* has two types of connectors for material flows and thus for connection of pipes.

- **N1 ... Nn – general connecting pipes**
  These connection points are used to fill and empty the pressure strainer and as gas connections.
  All connectors of the vector *N1..Nn* are initially equal and can be used for all connection types (water/steam, ideal gas, liquid).
- **F1 ... Fn – Filtrate drain**
  These connection points are a special case. The filtrate exits the system through these connections. All other connections are made using the vector *N1..Nn* described above.

The number of connection points depends on the system to be simulated and is specified using the parameters *NbrOfConnectors* for *N1 ... Nn* and *NbrOfFiltrateConnectors* for *F1 ... Fn*.

The medium in the neighboring flow networks is defined over the *NetParam* component.

The minimum connection is:

- Suspension entry (*Nn*)
- Filter cake discharge (fraction of mixture with low density) (*Nn*)
- Filtrate discharge (fraction of mixture with higher density) (*Fn*)
- Gas connector (*Nn*)

Possible additional connections include:

- Inflow/discharge of cleaning agents (*Nn*)
- Inflow/discharge of detergents (*Nn*)
- Additional gas inflow and outflow (*Nn*)

For the connector type Water/Steam, the inflowing medium is mixed with a liquid medium inside the pressure strainer. The inflowing mass and enthalpy is taken into account but the model does not form a corresponding phase equilibrium. Incoming steam is added to the mass balance of the liquid phase and does not contribute to pressure build-up in the gas phase. Water is handled in the mixture like a liquid medium.

If water/steam is used as a discharge medium, the mixture enthalpy of the liquid phase is specified at the corresponding material output of the pressure strainer. All other calculations within this flow network are made using the properties of water/steam (e.g. heat capacity).

Due to the lack of steam properties, it is not possible to model e.g. a steam sterilization, even at shutdown.

Parameter ConnectorHeight

For each of the connection points *N1..Nn* and *F1..Fn* you need the installation height of the nozzle on the pressure strainer (parameter *ConnectorHeight* for connectors *N1..Nn* or *ConnectorHeightF* for connectors *F1..Fn).* The information is entered in meters. The default is 0 m and thus at the bottom of the pressure strainer. Negative values are permitted for simulation of connectors at a lower position.

Input vector SolidFraction

The solids fraction of the suspension is specified accordingly at the input of the connectors *N1..Nn* and balanced in the pressure strainer. The balance calculation and parameter assignment of the component is the same as for the *StorageTank*.

Connectors without separation fractions (e.g. detergents) are given the default value "0". Gas connectors are also given the value "0".

Examples:

- Product suspension – e.g. 90%
- Solvent: 0%
- Detergent: 0%

The values defined here only apply to incoming substance flows. All outgoing material flows exit with the solids concentration currently stored in the pressure strainer. This value is available to the user at the *Rate* output for additional calculations. Any entered *SolidFraction* is not evaluated in this case.

The filtrate discharges *F1..Fn* are an exception once again. These outflows are always pure filtrate. This means there are no connectors *SolidFraction* for the filtrate discharges *F1..Fn*. If these connectors are used as inputs, balancing takes place with the *SolidFraction* = 0.0 (without separable components).

The input vector *SolidFraction* is not visible by default.

Alternatively, you can also manually set the *SolidFraction* parameter of the **filled pressure strainer** after the filling operation via the operating window.

Balance calculation of the liquid phase

All defined inputs in the input vector *N1 ..Nn* and *F1..Fn* must be connected. None of these inputs may remain open during simulation because this will create open flow networks. This will result in an error message during code generation.

The liquid phase is balanced with respect to the incoming and outgoing mass flows and enthalpies. In addition, a mass balance of the separable mixture component is carried out.

$$\frac{dm_{Liquid}}{dt} = \sum_{i=1}^{n} \dot{m}_{Liquid,i}$$

$$\frac{dm_{Screening}}{dt} = \sum_{i}^{n} \dot{m}_{Screening,i}$$

$$\frac{dh_{Liquid}}{dt} = \frac{1}{m_{Liquid}} \left(\sum_{i}^{n} \dot{m}_{Liquid,i}\, h_{Liquid,i} + \sum \dot{Q} \right)$$

$$T_{Liquid} = \frac{h_{Liquid}}{c_P}$$

Gas balance calculation

As soon as at least one gas connector is connected (characterized by a corresponding *NetParam* component), a corresponding gas pressure is also calculated. The gas pressure is changed by adding or removing gas and/or changing the gas volume (e.g. by filling the tank). The calculated gas pressure acts on the produced filter cake and presses the filtrate out of the filter cake until it reaches a configured level of dryness *DryingRate*.

As soon as the solids concentration currently stored in the tank *Rate* corresponds to the configured dryness level *DryingRate*, a pressure breakthrough of the filter cake occurs. The gas pressure in the pressure strainer drops abruptly. This pressure drop is indicated in the symbol by a color change of the cross-hatched filter zone to red.

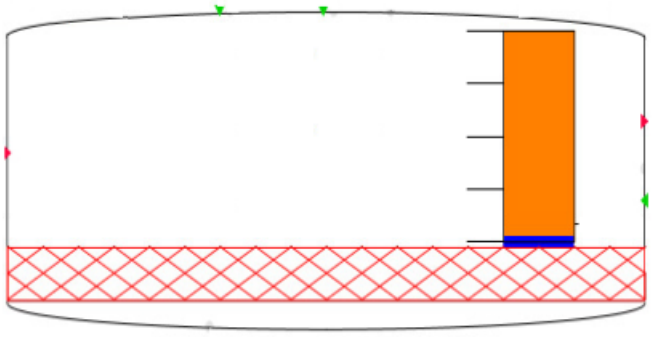


Figure 8-77 Symbol of the PressureStrainer upon reaching the configured dryness level DryingRate (pressure breakthrough)

The gas balance calculation only refers to externally added (inert) gases, however. Vapor pressures of solvents are not taken into consideration.

The incoming and outgoing mass flows and enthalpies are balanced within the vapor space.

$$\frac{dm_{Gas}}{dt} = \sum_{i}^{n} \dot{m}_{Gas,i}$$

$$\frac{dh_{Gas}}{dt} = \frac{1}{m_{Gas}} \left(\sum_{i}^{n} \dot{m}_{Gas,i}\, h_{Gas,i} + \sum \dot{Q} \right)$$

The gas pressure is then calculated according to the state equation of ideal gases:

$$p_{Gas} = \frac{m_{Gas} R_S T_{Gas}}{V_{Gas}}$$

The gas temperature T_{Gas} is calculated from the balanced gas enthalpy h_{Gas}. If the gas temperature and liquid temperature are different, a corresponding temperature compensation occurs during simulation.

The absolute value of the filtrate mass flow can be set by means of k_V values in the filtrate drain (e.g. valve or nozzle). An additional component is required to do this.

Overwriting state variables

By using the operating window of the component, you can set the following state variables to any (permissible) value during simulation:

- Stored mass / level
- Temperature
- Current solids concentration

For this, you must first enter the setpoint in the corresponding digital input *SetValues* and apply it with the "Set" button. You can specify the state of the stored mass using the "Mass" value as well as using the level (*Level*).

By manipulating state variables in this way, you can, for example, speed up heating processes or filling processes. As the operator, you can also empty the pressure strainer, for example, to return to an earlier step in a step sequence.

Removal of the solids

Removal of the solids from the pressure strainer by means of screw conveyors, scrapers or similar is not possible. The reason being is that while the solids fraction can be balanced as solid matter, it is handled internally in the model like a liquid. As a result, appropriate valves and pumps must be used for removal and operated by the trigger signals of the actual removal devices.

Borderline case "Pressure strainer empty"

Inflows into the pressure strainer and outflows out of the pressure strainer are generally throttled at the nozzle of the strainer. During normal operation, the user does not immediately recognize this process, because throttling is to have very little impact on the simulation. The outflow for an empty strainer is severely throttled. The flow coefficient is therefore set to maximum throttle for all connections over which a medium flows out.

These flow coefficients are calculated internally as a function of the tank volume and cannot be influenced by the modeler.

The pressure strainer is considered empty when the liquid amount M_W/ρ is less than a specified minimum tank level *RemainderLiquid*:

$m_L < RemainderLiquid \cdot \rho$

Balance calculation of the states is stopped for an empty strainer, which means changes to the mass, partial masses and specific enthalpy are discarded. The status "empty" only changes when the amount of liquid increases sufficiently, which means the difference between inflowing and outflowing quantity is positive.

Borderline case "Pressure strainer full"

The pressure strainer is considered full when its amount of liquid has reached the maximum value:

$m_L < (V - RemainderGas) \cdot \rho$

The balance calculation of the states is stopped for a full strainer That is, if the stored mass, partial mass or specific enthalpy is changed, the changes are discarded. The status "full" only changes when the amount of liquid decreases sufficiently, which means the difference between inflowing and outflowing quantity is negative.

Parameters

Parameter name	Description	Unit	Default value
Volume	Volume *V* of the tank; can be adjusted online	m^3	1.0
Height	Height of the tank; can be adjusted online	m	1.0
NbrOfInletsOutlets	Number *N* of connectors general	–	3
NbrOfFiltrateOutlets	Number *N* of filtrate connectors	–	1
NbrOfMeasurements	Number *N* of measuring points	–	1
MeasurementHeight [Nr]	Height of the measuring points above the tank bottom	m	0.0
ConnectorHeight [Nr]	Height of the general pipe connectors above the tank bottom	m	0.0
ConnectorHeightF [Nr]	Height of the filtrate connectors above the tank bottom	m	0.0
Boiling	Vapor pressure curve of the low boiler: Pressure in bar as function of the temperature (°C)	–	
Vaporization	Evaporization enthalpy of the low boiler in kJ/kg as a function of temperature (°C)	–	

Additional parameters

Parameter name	Description	Unit	Default value
PressureOutside	Ambient pressure	bar	1.0
LevelInit	Initial level	%	3.0
TemperatureInit	Initial temperature	°C	20.0
PressureInit	Initial pressure	bar	1.0
InitSolidPart	Initial mass fraction of solids in the suspension	%	0.0
Density	Mixture density in the tank	kg/m^3	997.337

Parameter name	Description	Unit	Default value
DryingRate	Maximum achievable solids concentration in the filter cake after which pressure breakthrough occurs	kg/kg	0.95
RemainderGas	Residual volume that cannot be filled with liquid when filling the strainer.	m^3	0.05
RemainderLiquid	Residual volume that remains in the pressure strainer after the strainer is emptied	m^3	0.005
geoHeight	Geodetic height; can be changed online	m	0.0

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit	Can variable be changed?
Length	L	m	Yes[1]
Temperature (fluid)	T_L	°C	Yes[1]
Temperature (gas)	T_G	°C	Yes[1]
Weight	M	kg	Yes[1]
Pressure	p_B	bar	No
Gas pressure	p_G	bar	No
Percentage of solids in the suspension	w	%	Yes[1]

[1] Enable input with "Set"

Connection example

The figure below shows an example for connection of a pressure strainer while using components of the CHEM-BASIC library.

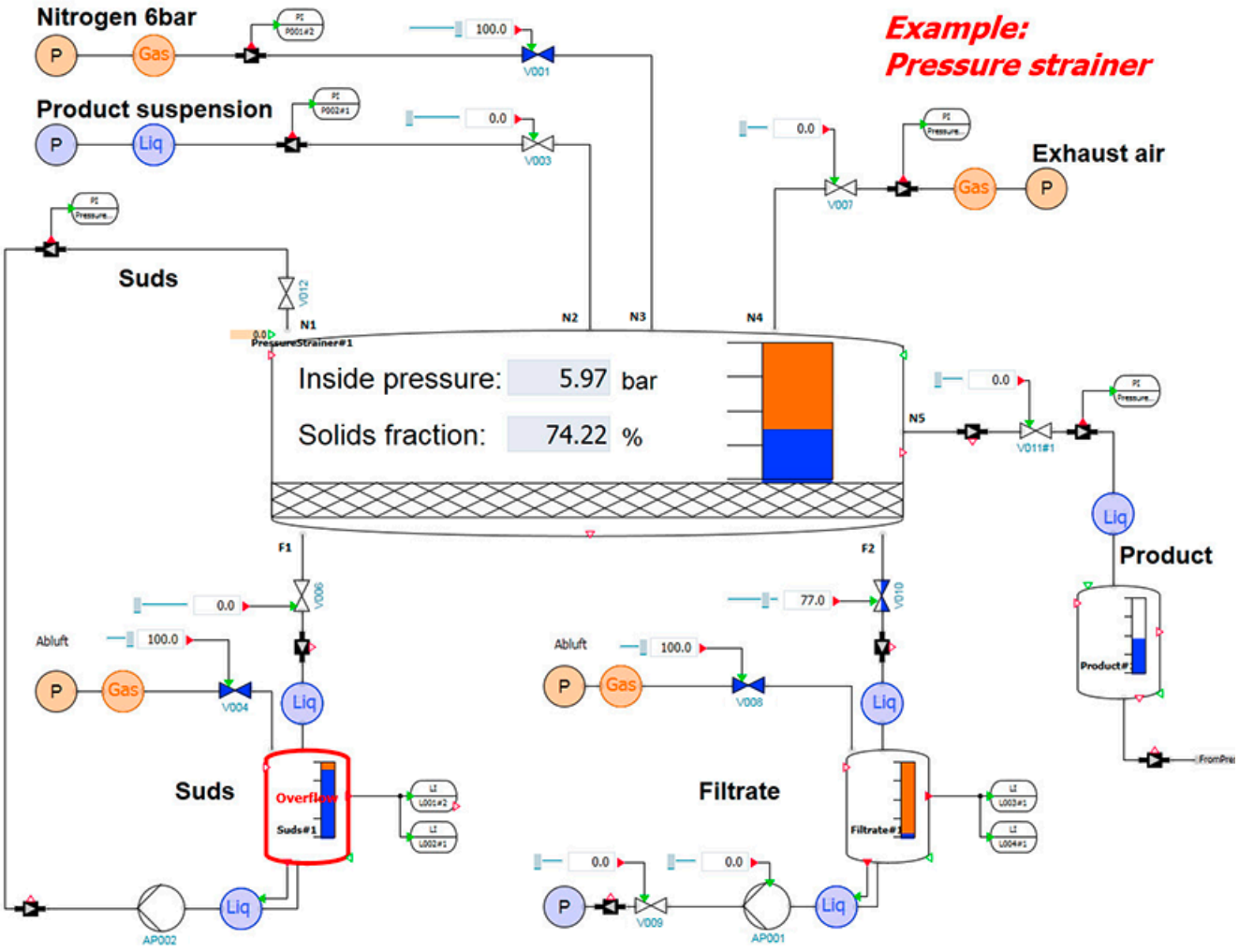


Figure 8-78 Connection example of component type PressureStrainer

The following parameter assignment of the *PressureStrainer* component is used for this connection example:

PressureStrainer#1

	Name		Value
General	Volume	[m³]	1.0
Input	Height	[m]	0.5
Output	NbrOfInletsOutlets		5
Parameter	NbrOfFiltrateOutlets		2
Additional parameter	NbrOfMeasurements		1
State	▾ MeasurementHeight [1]		...
	MeasurementHeight1	[m]	0.0
	▾ ConnectorHeight [5]		...
	ConnectorHeight1	[m]	1.0
	ConnectorHeight2	[m]	1.0
	ConnectorHeight3	[m]	1.0
	ConnectorHeight4	[m]	1.0
	ConnectorHeight5	[m]	-1.0
	▾ ConnectorHeightF [2]		...
	ConnectorHeightF1	[m]	0.0
	ConnectorHeightF2	[m]	0.0
	Boiling		
	Vaporization		

Figure 8-79 Parameter assignment of component PressureStrainer for the connection example

See also

Geodetic height (Page 618)

Screening device

Symbol

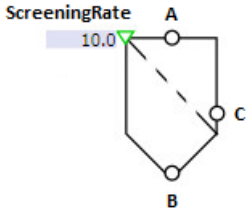


Figure 8-80 Symbol of component type ScreeningDevice

Function

The component type *ScreeningDevice* is used to separate a partial mass flow from a total mass flow.

Example:

- Separation of a specific particle size from a total batch
- Separation of a solids fraction from a suspension

This component has a screening stage.

The separable fraction of the total mass flow is specified using the input *RateA*.

The flownet of the separated partial mass flow *C* is topologically not connected to the flownet of the total mass flow *AB*. It must therefore be described using a separate *NetParam* block.

A separation can only take place when the flow network connected to the connector *C* is open.

During operation, at least the separable fraction must always be separated from the total mass flow. If this fraction cannot be discharged using the flow network connected to connector *C*, the main flow is reduced. The screen is blocked.

A corresponding flow coefficient of the screen (k_{VS} value) is also required for the main mass flow. If this flow coefficient is too small, the required mass flow may not be achieved.

The *Rate* output shows the percentage of the separable mixture component at output *B*. We assume a complete separation in the *Separator* component so that this percentage is always 0%.

There is no backflow via the connector *C*. A backflow from *B* to *A* is also not possible.

A pressure drop is calculated in the component for the main mass flow. This pressure drop depends on:

- Flow coefficient of the screen (k_{VS} value)
- Throughflow (volume flow, medium)

The pressure drop across the screen is calculated according to the following formula:

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_{VS}^2 \, \rho \frac{\text{kg}}{\text{m}^3}} \, 12960 \left(\frac{\text{sec}}{\text{h}} \right)^2$$

Whereby:

$\Delta p = p_B - p_A$	the pressure drop across the screen in bar
$\dot{m}$	the flow or mass flow in kg/s
ρ	the density of the medium in kg/m³
k_{VS}	the flow coefficient of the screen in m³/h

NOTICE
The flow network of the discharged mass flow at connector C begins in the component with a mass flow source. Because of this mass flow source, the pressure reached at the edge of the flow network is dependent on the separated mass flow and the pressure drop in the following branch. Pressure drops that are too high in the following branch (for example too low k_{VS} values in the valves) can result in impossible physical states (e.g. pressure in the output is higher than in the main flow).

Parameters

Parameter name	Description	Unit	Default value
Kvs	Flow coefficient k_{VS} with $k_{VS} \geq 10^{-6}$ m^3/h	m^3/h	10.0
geoHeight	Geodetic height; can be changed online	m	0.0

Operating window

The following variables for the following mass flows are displayed in the operating window:

- Incoming total mass flow
- Outgoing total mass flow
- Separated partial mass flow

Variable	Symbol	Unit
Mass fraction	w	%
Mass flow	$\dot{m}$	kg/s

See also

Geodetic height (Page 618)

Separator

Symbol

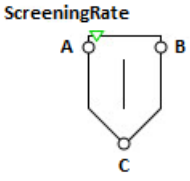


Figure 8-81 Symbol of component type Separator

Function

The separator is used for partial or complete separation of a partial mass flow *C* (e.g. of a solid mass fraction) from the main flow *AB*.

Because the flow network in SIMIT generally only permits the transport of gases and liquids, solids are handled like liquids here. A pressure difference is therefore necessary for solids transport. The conveyor is replaced with equivalent connections that consist of pumps, valves, sources and sinks.

The flow network of the separated partial mass flow *C* is topologically not connected to the flow network of the main mass flow *AB*. The flow network of the partial mass flow must therefore be described using a separate *NetParam* block.

A separation can only take place when the flow network connected to the connector *C* is open. For example, if this flow network is blocked by a closed valve, this separator does not perform any separation. In this case, the corresponding mass flow is forwarded to the output *B*.

A corresponding pressure drop is also required from the main mass flow to the connection point *C*.

Backflow via the connector *C* is not possible.

Depending on the flow direction, the mass fraction of the partial mass flow to be separated is specified using the input *RateA* or *RateB*.

A node is included in the *Separator* component. You can simulate a storage volume using the dynamics of this node. You must make node settings separately for the main flow network and the partial flow network using the corresponding network parameter assignment component (*NetParam*).

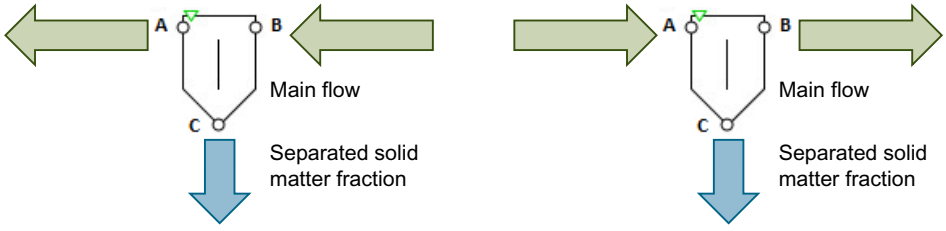


Figure 8-82 Possible flows of component type Separator

The main flow connectors *A* and *B* are designed identically. The partial mass flow to be separated is always removed via the connector *C* .

The mass fraction of the main flow exiting at *A* or *B* is calculated and displayed at the output *Rate*.

A pressure drop is calculated for the main mass flow. This pressure drop depends on:

- Pressure drop value of the separator (k_{VS} value)
- Throughflow (volume flow, medium)

The pressure drop across the separator is calculated according to the following formula:

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_{VS}^2 \, \rho \frac{\text{kg}}{\text{m}^3}} 12960 \left(\frac{\text{sec}}{\text{h}} \right)^2$$

Whereby:

$\Delta p = p_B - p_A$	the pressure drop across the separator in bar
$\dot{m}$	the flow or mass flow in kg/s
ρ	the density of the medium in kg/m³
k_{VS}	the flow coefficient of the separator in m³/h

Parameters

Parameter name	Description	Unit	Default value
Kvs	Flow coefficient k_{VS} with $k_{VS} \geq 10^{-6}$ m^3/h	m^3/h	10.0
geoHeight	Geodetic height; can be changed online	m	0.0

Operating window

The following variables for the following mass flows are displayed in the operating window:

- Incoming main mass flow
- Outgoing main mass flow
- Separated partial mass flow

Variable	Symbol	Unit
Mass fraction	w	%
Mass flow	$\dot{m}$	kg/s

See also

Geodetic height (Page 618)

Settler

Symbol

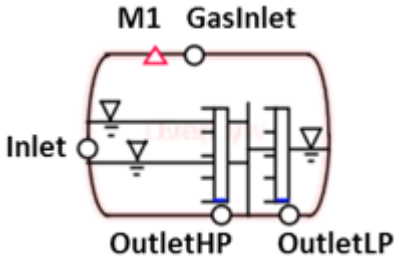


Figure 8-83 Settler

Function

The component type *Settler* calculates the separation of liquid media that form two phases. A liquid flow enters the horizontal cylindrical settler via connector *Inlet*. In the inlet area (*SettlingArea*), the two phases are separated into a light phase (*LightPhase*) and a heavy phase (*HeavyPhase*). The light phase fraction is specified using *LightPhaseFractionIn* at the inlet of the *Settler*. The heavy phase settles at the bottom and the light phase settles at the top. This happens immediately in the component. Any time needed for mapping the settling behavior is disregarded The settler contains a weir. The horizontal position of the weir can be specified using the *WeirPosition* parameter. The weir height is specified with the *WeirHeight* parameter. If the level of the settling area is above the weir height, liquid flows over the weir into the light area (*PureLightPhase*). Both the light phase and heavy phase can flow over the weir depending on the height of the phase boundary interface, i.e. the level of the heavy phase in the *SettlingArea*. The outlets (*OutletHP* and *OutletLP*) are at the bottom of the settler. Outlet *OutletHP* belongs to area *SettlingArea*, while outlet *OutletLP* belongs to area *PureLightPhase*. Because the heavy phase always settles at the bottom, it is always removed first from the system via the outlets. The liquid phases can have a residual solubility for the other phase in each case, however. The residual solubility of the heavy phase in the light phase is specified with the *SolubilityHP* parameter and the residual solubility of the light phase in the heavy phase with the *SolubilityLP* parameter.

For the *SettlingArea* and *PureLightPhase* areas, the incoming and outgoing mass flows are automatically balanced separately from one another with respect to mass and enthalpy. A temperature control of the mass accumulated in the tank is possible using the direct input *Q*. The mass balanced in the tank is a mixture of a maximum of two liquid components. Liquid can only flow into the settler via connector *Inlet*. Liquid can only flow out of the settler via connectors *OutletHP* and *OutletLP*. The reverse flow direction in each case is not possible. A gas network is connected using connector *GasInlet*. The pressure in the settler is specified by means of the gas network by pressing on the liquid or drawing a vacuum.

The media in the adjoining flow networks are defined using the *NetParam* component. Water/steam is thereby treated as a liquid medium. All connectors must be interconnected. None of these inputs may remain open during simulation because this would create open flow networks. This would, in turn, result in an error message during code generation.

Borderline case "empty separator"

Inflows into the settler and outflows from the settler are generally throttled at the nozzle of the settler. During normal operation, this throttling is not immediately recognizable for the user, because throttling should have very little impact on the simulation. For an empty settler, the outflow becomes strongly throttled. The flow coefficient is therefore set to the value for maximum throttling for all connectors via which liquid flows out. These flow coefficients are internally calculated as a function of the volume and cannot be influenced by the user. The settler is considered empty when the liquid quantity is less than a specified minimum settler level in percent *MinLiquidVolumePercent*:

The balancing of the states is stopped for an empty settler. That is, changes of mass, partial masses and specific enthalpy are discarded. The two chambers of the settler are examined separately. The "empty" state continues until there is a sufficient increase in the liquid quantity. That is, the result of inflowing quantity minus outflowing quantity is positive. A corresponding indication of an empty settler is shown in the component symbol.

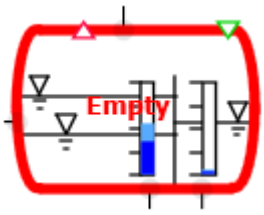


Figure 8-84 Empty settler

Borderline case "full settler"

The settler is considered to be full when its liquid quantity reaches the maximum possible percentage value and the gas quantity is thus less than a given minimum settler level in percent *MinGasVolumePercent*.

The balancing of the states is stopped for a full settler. That is, changes of accumulated mass, partial mass and specific enthalpy are discarded. The "full" state continues until there is a sufficient decrease in the liquid quantity. That is, the result of inflowing quantity minus outflowing quantity is negative. A corresponding indication of a full settler is shown in the component symbol.

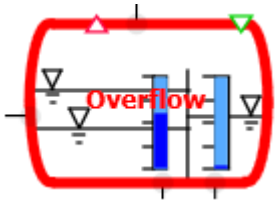


Figure 8-85 Full settler

Parameters

Parameter name	Description	Unit	Default value
Volume	Volume V of the settler	m^3	1.0
Length	Length of the settler	m	2.0
NbrOfMeasurements	Number N of process tags	-	1
MeasurementHeight	Height of the process tags above the tank bottom	m	0.0
WeirHeight	Height of the weir	m	0.4
WeirPosition	Distance of the weir from the inlet in m	m	1.5

Additional parameters

Parameter name	Description	Unit	Default value
LevelInit	Initialization level	%	50.0

TemperatureInit	Initialization temperature	°C	20.0
PressureInit	Initialization pressure	bar	1.0
InitLightPhase	Initialization mass fraction of light phase	%	0.0
DensityLP	Density of light phase	kg/m³	800.0
DensityHP	Density of heavy phase	kg/m³	1000.0
SolubilityLP	Residual solubility of the light phase in the heavy phase	kg/kg %	0.0
SolubilityHP	Residual solubility of the heavy phase in the light phase	kg/kg %	0.0
MinGasVolumePercent	Percentage of tank volume that cannot be filled with liquid when filling the tank	%	5
MinLiquidVolumePercent	Percentage of tank volume that remains in the tank when draining the tank	%	0.5
geoHeight	Geodetic height of the tank	m	0.0

Operating window

Variable	Symbol	Unit	Remark
Level	L	m	For SettlingArea area and PureLightPhase area, respectively
Level of heavy phase	LHP	m	For SettlingArea area and PureLightPhase area, respectively
Mass	M	kg	For SettlingArea area and PureLightPhase area, respectively
Mass fraction of light phase	wLP	%	For SettlingArea area and PureLightPhase area, respectively
Temperature of liquid	T	°C	Global
Gas pressure	PG	bar	Global

VacuumDryer

Symbol

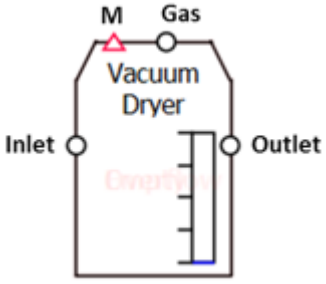


Figure 8-86 VacuumDryer

Function

The component type *VacuumDryer* calculates the drying of a mass flow by drawing a vacuum. The incoming and outgoing mass flows are balanced with respect to mass and enthalpy. In addition, a mass balance of the moisture is carried out. A direct input can be used for temperature control of or add heat to the accumulated mass being dried in the dryer.

The material to be dried is fed to the *VacuumDryer* via connector *Inlet*. The dried material exits the dryer via connector *Outlet*. A flow network with gas medium must not be connected to connectors *Inlet* and *Outlet*. The moisture is removed from the dryer via connector *Gas*. A flow network with the gas or water/steam medium must be connected to this connector. In the case of water/steam, the process conditions in the *VacuumDryer* must be such that the water is in vapor form.

The medium in the adjoining flow networks is defined using the *NetParam* component. All defined inputs in the input vector *N1...Nn* must be interconnected. None of these inputs may remain open during simulation because this would create open flow networks. This would, in turn, result in an error message during code generation.

Evaporation

To evaporate the moisture, the material that is being dried must be heated to a temperature that is above the boiling point corresponding to the current pressure.

The vapor pressure curve needed for this is configured using the *VaporPressureMoisture* parameter (vapor pressure curve of low-boiler $p_S = f\ (T_S)$ in bar and °C). In addition, the enthalpy of vaporization of the low-boiler must be specified as a characteristic curve ($\Delta H_V = f(T_S)$ in kJ/kg and °C) using the *EnthalpyOfVaporization* parameter. These characteristic curves are preset for water. They can be edited within the charts or read in from a text file. Examples of these characteristic curves are shown in the figures below:

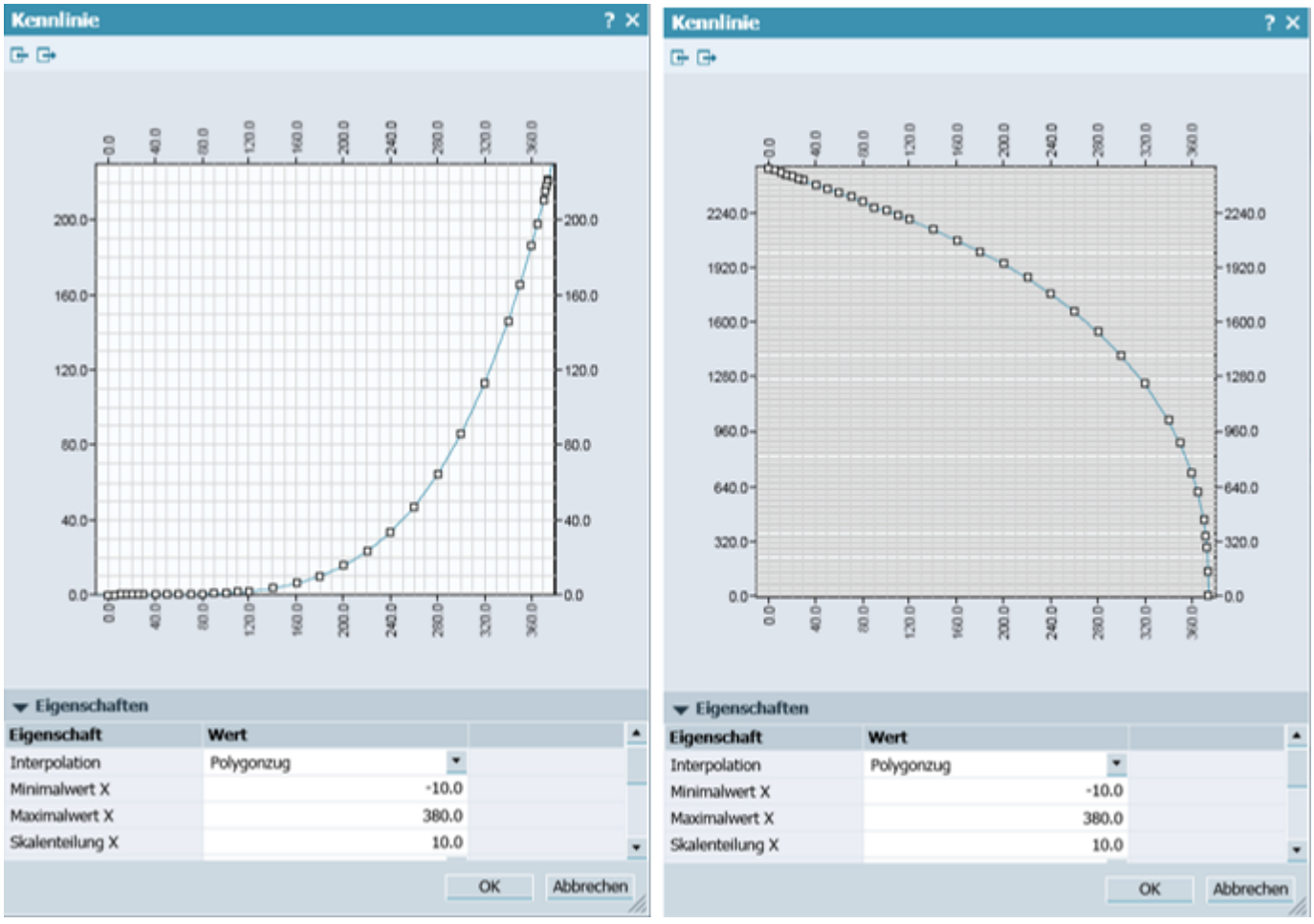


Figure 8-87 Example of enthalpy of vaporization and vapor pressure curve

When the evaporation condition $T > T_S$ (p_S) is reached, the low-boiler gradually evaporates to the vapor phase and is accounted for as vapor in the mass balance. No provision is made for condensation in the tank.

The evaporation quantity $\dot{m}_{Evap}$ is calculated as a function of the pressure difference ΔP between vapor pressure and pressure in the dryer and the temperature T:

$$\dot{m}_{Evap} = f_{Evap} \cdot \frac{\Delta P}{R_m \cdot T} \cdot A$$

f_{Evap}	Evaporation factor *EvapFactor*
R_m	Specific gas constant
A	Cross-sectional area of the tank

The *EvapRateVariability* parameter can be used to adapt the rate of change of the evaporation quantity. If values are too low, the state of equilibrium may be reached only very slowly or may not be reached at all. If values are too high, instability may result.

Borderline case "empty dryer"

Inflows into the dryer and outflows from the dryer are generally throttled at the nozzle of the dryer. During normal operation, this throttling is not immediately recognizable for the user, because throttling should have very little impact on the simulation. For an empty dryer, the outflow becomes strongly throttled. The flow coefficient is therefore set to the value for maximum throttling for all connectors via which liquid flows out. The flow coefficients are internally calculated as a function of the dryer volume and cannot be influenced by the modeler. The dryer is considered empty when the quantity of material to be dried is less than a specified minimum level in percent *MinLiquidVolumePercent*:

The balancing of the states is stopped for an empty dryer. That is, changes of mass, partial masses and specific enthalpy are discarded. The "empty" state continues until there is a sufficient increase in the liquid quantity. That is, the result of inflowing quantity minus outflowing quantity is positive. A corresponding indication of an empty dryer is shown in the component symbol.

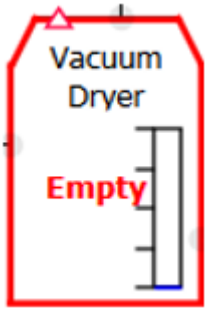


Figure 8-88 Empty dryer

Borderline case "full dryer"

The dryer is considered to be full when the quantity of material being dried reaches the maximum possible percentage value, and the gas volume is thus less than a given minimum dryer level in percent *MinGasVolumePercent*.

The balancing of the states is stopped for a full dryer. That is, changes of accumulated mass, partial mass and specific enthalpy are discarded. The "full" state continues until there is a sufficient decrease in the liquid quantity. That is, the result of inflowing quantity minus outflowing quantity is negative. A corresponding indication of a full dryer is shown in the component symbol.

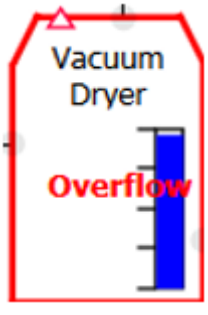


Figure 8-89 Full dryer

Parameters

Parameter name	Description	Unit	Default value
Volume	Volume V of the dryer	m^3	1.0
Height	Height of the vertical dryer or length or the horizontal dryer	m	1.0
VaporPressureMoisture	Vapor pressure curve of moisture: Pressure in bar as a function of temperature (°C)	-	
EnthalpyOfVaporization	Enthalpy of vaporization of moisture in kJ/kg as a function of temperature (°C)	-	

Additional parameters

Parameter name	Description	Unit	Default value
LevelInit	Initialization level	%	10.0
TemperatureInit	Initialization temperature	°C	20.0
PressureInit	Initialization pressure	bar	0.01
HeatCapacityInit	Initialization heat capacity	kJ/kgK	4.18
MoistureInit	Initialization mass fraction of moisture	%	0.0
BulkDensity	Density of heavy phase	kg/m^3	1000.0
EvapFactor	Evaporation factor	-	1.0
EvapRateVariability	Variability of evaporation rate	-	1.0
MinGasVolumePercent	Percentage of dryer volume that cannot be filled with liquid when filling the dryer	%	5
MinLiquidVolumePercent	Percentage of dryer volume that remains in the dryer when the dryer is emptied	%	0.5
geoHeight	Geodetic height of the dryer	m	0.0

Operating window

Variable	Symbol	Unit
Level	L	m
Mass	M	kg
Temperature	T	°C
Gas pressure	P_G	bar
Mass fraction of moisture	w_M	%

Mean residence time	t	s
Evaporation performance	Q_{Evap}	kW

8.2.5.10 System

BranchParam – Branch parameter assignment

Symbol



Figure 8-90 Symbol of component type BranchParam

Function

The component type *BranchParam* is used for parameter assignment of a branch in the flow network. For this purpose, the component is inserted anywhere in the branch whose parameters are to be assigned.

Parameters

Parameter name	Description	Unit	Default value
FactorMomentum	Momentum factor for the flow in the branches of the flow network	m	450.0

Mnode – Mass flow setting

Symbol



Figure 8-91 Symbol of component type Mnode

Function

The component type *Mnode* specifies the values for mass flow $\dot{m}$ and specific enthalpy h at its connector A. This type of component forms a boundary for the flow network. When looking at the flow network as a graph, this corresponds to an inflow or outflow into an (internal) node or branch. An internal node with defined inflow or outflow is added to the flow network for each component of this type.

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit	Can variable be changed?
Mass flow (Default: 0)	$\dot{m}$	kg/s	Yes
Specific enthalpy (Default: 100)	h	kJ/kg	Yes

The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Pressure	p	bar
Density	r	kg/m³
Temperature	T	°C
Specific enthalpy	h	kJ/kg

NetParam – Network parameter assignment

Symbol



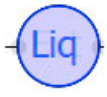
Figure 8-92 Symbol of component type NetParam

Function

The component type *NetParam* is used for parameter assignment of a flow network. To do so, the component is added at any location in any branch of the flow network.

Representation of the *NetParam* component depending on the set medium:







Medium	Gas	Liquid	Water/Steam

The parameters specified in this component apply to the entire flow network. They can be read and further processed by other components included in the flow network.

The *NetParam* component has the following inputs:

- *MediumExt*
  External switchover of the medium set in the flow network during runtime through events from the model. Defined inputs are:
  - –9999 (default): This input is not used, the medium set in the Medium parameter is current
  - 0: Switchover to "Water/Steam" medium
  - 1: Switchover to "Ideal Gas" medium
  - 2: Switchover to "Liquid" medium
- *CpExt*
  External input for heat capacity in the flow network (can, for example, be connected from the output from an upstream tank). Only values greater than zero are accepted as input; for input zero (default), the parameters *sHeatCapGas* or *sHeatCapLiquid* apply.

Both inputs have the default "not visible" and must be made visible before they can be used.

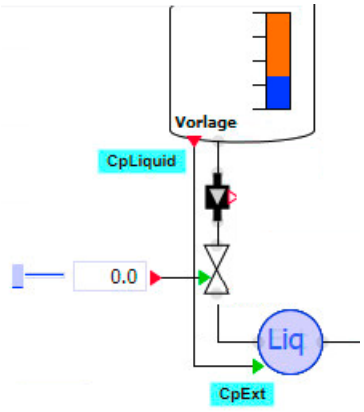


Figure 8-93 Example for the connection of the CpExt input

NOTICE
The component lets you switch the medium during runtime. However, this should only be done in exceptional cases because, depending on the connection of the components, the model may enter an undefined state. Model crashes or reaching of undefined states may be possible.

If no component is used in a flow network for parameter assignment of a flow network, the calculation is made with the defaults for water/steam.

Parameters

Parameter name	Description	Unit	Default value
Medium	For the medium in the flow network, you can select "Water/Steam", "Liquid" or "Ideal Gas".	–	Liquid
FactorMomentum	Momentum factor for the flow in the branches of the flow network	m	450.0
sCompressionGas	Specific compression module for "Water/Steam" medium with densities ρ < 500 kg/m³ or "Ideal Gas" medium	bar/kg	10.0
sCompressionLiquid	Specific compression module for "Water/Steam" medium with densities ρ > 500 kg/m³ or "Liquid" medium	bar/kg	100.0
FactorThermalGas	Factor for the enthalpy balance calculation of "Water/Steam" medium with densities ρ < 500 kg/m³ or "Ideal Gas" medium	1/kg	100.0
FactorThermalLiquid	Factor for the enthalpy balance calculation of "Water/Steam" medium with densities ρ > 500 kg/m³ or "Liquid" medium	1/kg	0.1
DensityLiquid	Media density; only valid for "Liquid" medium	kg/m³	997.337
sHeatCapGas	Specific heat capacity for gas; only valid for "Ideal Gas" medium	kJ/kgK	1.0
sHeatCapLiquid	Specific heat capacity for liquids; only valid for "Liquid" medium	kJ/kgK	4.18
GasConstant	Specific gas constant of the medium; only valid for "Ideal Gas" medium	kJ/kgK	0.287

Additional parameters

Parameter name	Description	Unit	Default value
PressureInit	Initialization value for the pressure in the internal nodes of the flow network	bar	1.0
sEnthalpyInit	Initialization value for the specific enthalpy in the internal nodes of the flow network	kJ/kg	100.0
SmoothTransition	When this additional parameter is set to True, the variables *sCompression* and *FactorThermal* are calculated with a density-dependent linear transfer function	–	False
TemperatureEnvironment	Ambient temperature	°C	20.0
FactorHeatExchangeEnv	Factor of proportionality for heat exchange of the medium in the internal nodes with the environment; no heat exchange occurs when the value zero is set	kW/kgK	0.0

PHnode – Pressure and enthalpy default

Symbol



Figure 8-94 Symbol of component type PHnode

Function

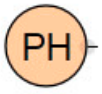
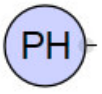
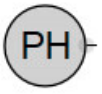
The component type *PHnode* represents an option for closure of a flow network to the outside. A fixed pressure *p* and the associated enthalpy *h* are specified as a boundary condition for the flow network in the component.

When looking at the flow network as a graph, this component corresponds to an (external) node for which the entered conditions are permanently specified.

Display

During the simulation, the selected medium is displayed in color in the symbol of the component. The component receives the information required for this display from the associated flow network parameter assignment (*NetParam* component).

Representation of the *PHnode* component depending on the set medium:

Medium	Gas	Liquid	Water/Steam
			

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit	Can variable be changed?
Pressure (Default: 1)	p	bar	Yes
Specific enthalpy (Default: 100)	h	kJ/kg	Yes

The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Mass flow[1]	$\dot{m}$	kg/s
Density	r	kg/m³

Variable	Symbol	Unit
Temperature	T	°C
Specific enthalpy	h	kJ/kg

[1] For outflow $\dot{m} > 0$, for inflow $\dot{m} < 0$.

PTnode – Pressure and temperature default

Symbol



Figure 8-95 Symbol of component type PTnode

Function

The component type *PTnode* represents an option for closure of a flow network to the outside. A fixed pressure *p* and the associated temperature *T* are specified as a boundary condition for the flow network in the component.

When looking at the flow network as a graph, this component corresponds to an (external) node for which the entered conditions are permanently specified.

Note

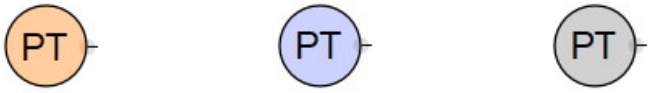
Please note that you cannot enter a state in the two-phase area with the medium "Water/Steam". The state is either completely liquid or completely vaporous depending on the temperature and pressure. If the boiling point is entered at the relevant pressure, the state is completely gaseous.

If you want to specify a state in the two-phase area, you will need to use component type *PHNode*.

Display

During the simulation, the selected medium is displayed in color in the symbol of the component. The component receives the information required for this display from the associated flow network parameter assignment (*NetParam* component).

Representation of the *PTnode* component depending on the set medium:



Medium Gas Liquid Water/Steam

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure	p	bar
Temperature	T	°C

The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Mass flow	$\dot{m}$	kg/s
Density	r	kg/m³
Temperature	T	°C
Specific enthalpy	h	kJ/kg

8.2.5.11 Tanks

Agitator

Symbol



Figure 8-96 Symbol of component type Agitator

Function

The component type *Agitator* exists to emulate the presence and operation of an agitator. The agitator is switched on via the *Speed* connector. The rotation of the agitator is shown graphically at a *Speed* > 0.

However, this component has no effect on mixing in a stirring tank because stirring tanks are always assumed to be ideally mixed systems.

This component also has no effect on the heat transfer from a heating element to a stirring tank. To model the heat transfer, you must link the agitator speed with the *HeatingJacket* component.

DrumWS – Storage tank for water/steam

Symbol

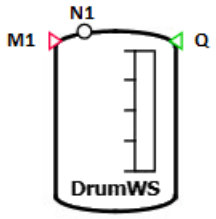


Figure 8-97 Symbol of component type DrumWS

Function

The component type *DrumWS* provides the simulation of a water/steam tank that is closed off from the environment.

The closed tank is assumed to be a horizontal or vertical cylindrical tank whose inflows and outflows are via connectors *N1* .. *Nn*. A throttling action is assumed for each connector according to the following formula:

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_{VS}^2 \, \rho \frac{\text{kg}}{\text{m}^3}} \, 12960 \left(\frac{\text{sec}}{\text{h}} \right)^2$$

The installation height of the individual connectors is specified for each connector with the *ConnectorHeight* parameter vector. The type of each connector (inlet or outlet, water or steam) is defined dynamically during the simulation and depends on the thermodynamic states in the tank, the configured geometry (*ConnectorHeight*, *Orientation*) and the respective boundary conditions.

You can move the connectors as needed along the outline of the component symbol with the mouse while pressing the <Alt> key.

An immediate separation of the medium into two homogeneous saturated phases is assumed in the tank: the saturated liquid phase and the saturated vapor phase. Only connectors to water/steam-type flow networks are included in the balance calculation. Connectors to other types of flow networks (gas, liquid or user-specific media) are not balanced.

This way you obtain the mass of water and steam in the tank that is balanced using the mass flows according to

$$\frac{dm}{dt} = \sum_{i=1}^{n} \dot{m}_i$$

or due to

$m = \rho V$

the change of the average density to

$$V\frac{d\rho}{dt}=\sum_{i=1}^{n}\dot{m}_i$$

The energy is further balanced according to

$$\frac{d(hm)}{dt}=\sum_{i=1}^{n}h_i\dot{m}_i$$

Whereby h is the specific enthalpy (mean enthalpy) of water/steam in the tank and h_i is the specific enthalpy of the inflow and outflow at the i-th connector. For outflows, the specific enthalpy of the medium in the tank is to be used, which means h' for outflows of saturated water, h'' for outflows of saturated steam. The balancing is calculated as follows

$$m\frac{dh}{dt}=\sum_{i=1,i\in Z}^{n}(h_i-h)\dot{m}_i+(h'-h)\sum_{i=1,i\in A'}^{n}\dot{m}_i+(h''-h)\sum_{i=1,i\in A''}^{n}\dot{m}_i$$

In this case, the inflows Z, outflows of saturated water A' and outflows of saturated vapor A'' are added up.

To model a heat exchange with the ideally insulated tank, the enthalpy balance is added according to

$$m\frac{dh}{dt}=\sum_{i=1,i\in Z}^{n}(h_i-h)\dot{m}_i+(h'-h)\sum_{i=1,i\in A'}^{n}\dot{m}_i+(h''-h)\sum_{i=1,i\in A''}^{n}\dot{m}_i+A\alpha(T_T-T_S)$$

T_T is the temperature of the tank wall, T_S the saturation temperature of water/steam.

The heat storage in the tank wall is described with the heat balance:

$$\frac{dT_T}{dt}=\frac{1}{m_Tc_T}A\alpha(T_S-T_T)$$

m_T Weight of the tank wall [kg]

c_T Heat capacity of the tank wall [kJ/(kg·K)]

The level of water l, the saturation temperature T_S, saturation pressure p_S and the mass m of water/steam are output at the *Measure* connector.

The connectors *N1* .. *Nn* are assumed according to the configured height. For connectors below the level, the pressure at these connectors is therefore made up of the saturation pressure and the gravitational pressure of the water as a function of the difference from the connector height (l_A) and level (l):

$p = \rho' g (l - l_A) + p_S$

l_A Height of the connection pipe [m]

Borderline case "empty tank"

The outflow for an empty tank is severely throttled. The flow coefficient is thereby set to the value k_{V0} for maximum throttling of all connectors through which water or steam flows out.

The tank is considered empty when the amount of water m_W/ρ is lower than a specified minimum tank filling V_{min}:

$m_W < V_{min}\rho$

The balancing of the states is stopped for an empty tank, which means changes to the density and specific enthalpy are discarded. The status "empty" only changes when the amount of water increases sufficiently. A check is therefore made in each time increment to see if

$$m_W \geq V_{min}\,\rho\left(1+\frac{H_{min}}{100\%}\right)$$

The required increase can be set with the hysteresis H_{min}.

A corresponding note is shown in the symbol of a component for an empty tank (see figure).

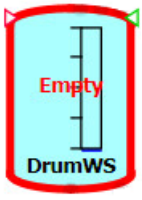


Figure 8-98 Display of an empty tank in the symbol

Borderline case "full tank"

The tank is considered full when the amount of water has reached the maximum value:

$m_W > (V - V_{min})\,\rho$

The balancing of the states is stopped for a full tank, which means changes to the density and specific enthalpy are discarded. The status "full" only changes when the amount of water decreases sufficiently. A check is therefore made in each time increment to see if

$$m_W \leq \rho V - \rho V_{min}\left(1+\frac{H_{min}}{100\%}\right)$$

The required decrease can be set with the hysteresis H_{min}.

A corresponding note is shown in the symbol of a component for a full tank (see figure).

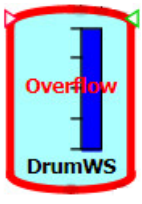


Figure 8-99 Display of a full tank in the symbol

Parameters

Parameter name	Description	Unit	Default value
Volume	Volume *V* of the tank; can be adjusted online	m^3	1.0
HeightOrLength	Height or length of the tank; can be changed online	m	1.0
NbrOfConnectors	Number of connectors N	–	1
NbrOfMeasurements	Number N of measuring points	–	1
MeasurementHeight [Nr]	Height of the measuring points above the tank bottom	m	0.0
ConnectorHeight [Nr]	Height of the pipe connectors above the tank bottom	m	0.0
Position	Tank position: *Vertically* or *Horizontally* If *Horizontally* is selected, a horizontal cylinder is assumed for the level.	–	*Vertically*

Additional parameters

Parameter name	Description	Unit	Default value
LevelInit	Initialization value for the level (water volume) in %	m^3	50.0
TemperatureInit	Initialization value for the temperature of water/steam (saturation temperature)	m	20.0
Kvs	Uniform flow coefficient k_V for all connectors	m^3/h	360.0
Kv0	Flow coefficient k_{V0} for severe throttling in the tank connector	m^3/h	0.000001
MinVolume	Minimum tank volume V_{min}; can be adjusted online	m^3	0.01
MinVolumeHys	Hysteresis H_{min}; can be adjusted online	%	50.0
MassDrum	Mass m_T of the tank; can be changed online	kg	5000.0
SurfaceDrum	Inner area *A* of the tank; can be changed online	m^2	12.5
sHeatCapDrum	Specific heat capacity c_T of the tank; can be changed online	kJ/kgK	0.5
HeatTransCoe	Heat transfer coefficient α for water/steam of the tank; can be changed online	kW/m^2K	0.0
geoHeight	Geodetic height; can be changed online	m	0.0

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit	Can variable be changed?
Water level	L	m	Yes[1]
Saturation temperature	T	°C	Yes[1]
Weight of water/steam	M	kg	Yes[1]
Saturation pressure	p_B	bar	No
Water pressure	p_G	bar	No
Container temperature	T_{Drum}	°C	No

[1] Enable input with "Set"

The following variables are displayed in the extended operating window:

- Mix: Intermediate variables
- Water: Variables for saturated water
- Steam: Variables for saturated steam

Variable	Symbol	Unit
Density	r	kg/m³
Specific enthalpy	h	kJ/kg

See also

Geodetic height (Page 618)

ElectricalHeatingJacket – Electrical heating jacket

Symbol

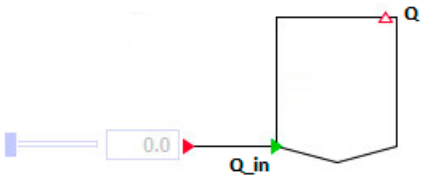


Figure 8-100 Symbol of component type ElektricalHeatingJacket

Function

The electrical heating jacket is an add-on to the *StorageTankLiquid*. The heating power at input Q_In is supplied as heat flow Q to the tank. The heat flow is specified in kW.

The component is used to visualize the heating jacket in the SIMIT chart. For simulation purposes, the heating power can also be connected directly to a *StorageTankLiquid*.

Unlike with the *HeatingJacket* component, the influence of the agitator speed or of the tank level is not taken into consideration here.

This component does not have parameters or an operating window.

Flash – 2-phase separator

Symbol

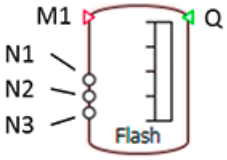


Figure 8-101 Symbol of component type Flash

Function

The *Flash* component describes a 2-phase separator and is used to separate the content into a liquid and gas phase. Depending on the temperature and pressure, the phase equilibrium is established between the low boiler (LB = Low Boiling) and the high boiler (HB = High Boiling). The incoming and outgoing mass flows are hereby automatically balanced regarding mass and enthalpy. Temperature control of the mass stored in the tank is possible by means of a direct input or various additional components.

The phase equilibrium is parameterized over the normal boiling temperature (at 1 bar) of the low and high boiler (*TS_LB*, *TS_HB*) as well as over their evaporization enthalpies (*hv_LB*, *hv_HB*). The size of the boiling lens can be varied using the *S* parameter. This means:

- $S = 0$, maximum size of the boiling lens (see figure)
- $S = T_{HB} - T_{LB}$, no boiling lens (makes no sense physically)

It is assumed that there is an equal pressure dependency of the boiling points for both materials. This is parameterized with the boiling temperature of the high boiler (*T2_HB*) for another (preferably higher) pressure (*p2_HB*).

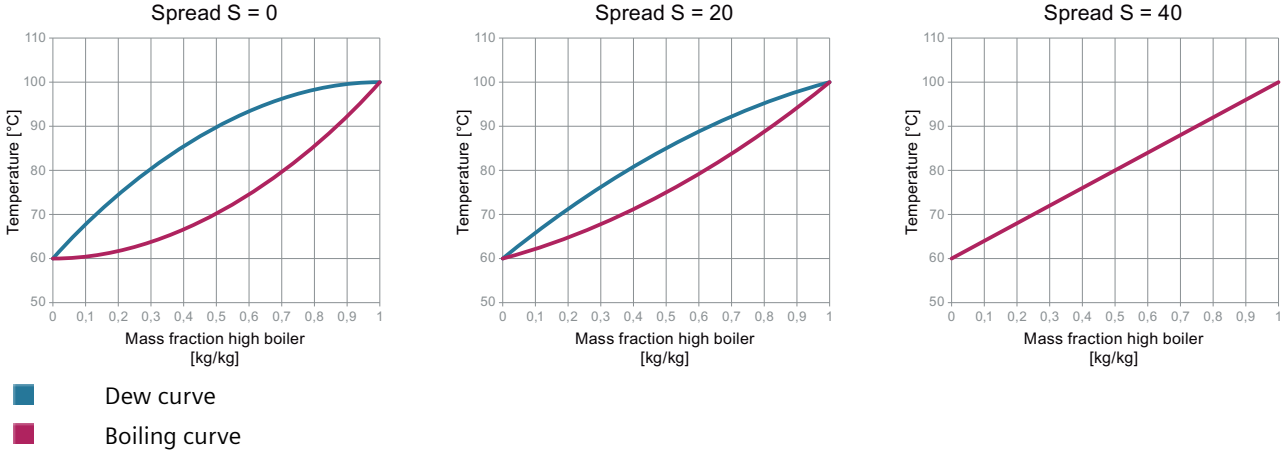


Figure 8-102 Boiling lens as a function of the parameter *S* (spread)

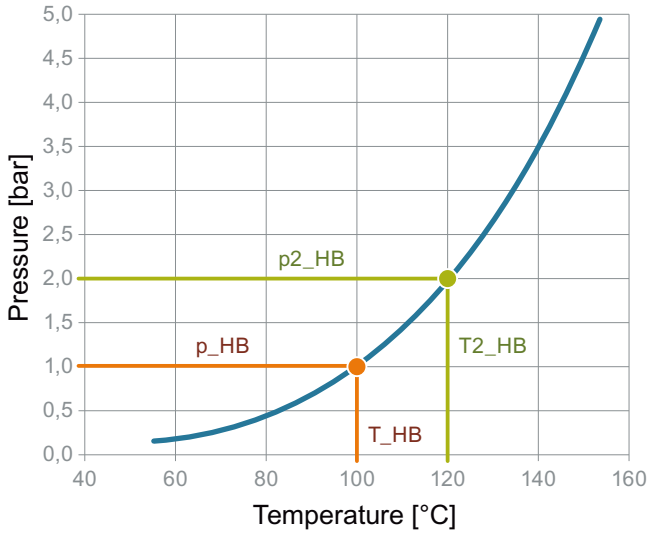


Figure 8-103 Dependency of the boiling temperature on the pressure

Circuit

The *N1 ..Nn* connectors are the connection points for the material flows, which means for the connection of pipes. The number of these connection points depends on the system to be simulated and is specified with the *NbrOfConnectors* parameter. All connectors of the input vector *N1..Nn* are initially equal and can be used for all connector types (water/steam, ideal gas, liquid).

The medium in the neighboring flow networks is defined over the *NetParam* component. All defined inputs must be interconnected in the input vector *N1 ..Nn*. These inputs must not remain open during simulation because this will create open flow networks. This would, in turn, result in an error message during code generation.

For the water/steam connector type, the incoming medium is mixed internally in the tank with a liquid medium. The incoming mass and enthalpy is taken into consideration.

A flash typically has three connectors. The feed comes in through the first connector. At the second connector on the top of the tank, the gas flow is removed. At the third connector at the bottom of the tank, the liquid drain is connected.

At least one gas flow must be connected to the flash.

Input vector *z_in* and output vector *z_aus*

The mass fraction of the light boiler in the incoming medium is specified with the input vector *z_in* for each of the connectors *N1..Nn*. Connectors without light boiler component are assigned the value "0". The specified values only apply to incoming material flows.

For all outgoing material flows, the mass fraction of the light boiler is transferred to the output vector *z_aus*. For gas connectors, the concentration of the gas phase is output. For all other connectors, the concentration of the liquid phase is output. In case of outgoing mass flows, a parameter entered in *z_in* may not be evaluated.

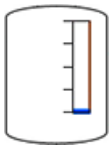
Parameter *ConnectorHeight*

The installation height of the connecting pipe at the flash is required for each of the connection points *N1..Nn* (*ConnectorHeight* parameter). The information is entered in meters. The default setting is 0 m, which means at the bottom of the tank. Negative values are permitted for the simulation of connectors at a lower position. If a gas connector is located below the level of the tank, neither liquid nor gas can flow out through this connector from the tank. However, the inflow of gas is possible.

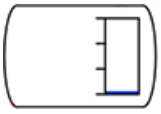
Orientation

The *Flash* component can be operated as vertical or horizontal tank. The orientation can be set with the *Position* parameter. The tank is simulated as either vertical or horizontal cylinder. In case of a vertical flash, the *HeightOrLength* parameter indicates the height of the tank; for a horizontal flash, it indicates the length.

Vertical flash



Horizontal flash



Heating / cooling of the flash tank

The flash can be heated or cooled with the following options:

- Combination with *HeatingJacket* component
- Combination with *ElectricalHeatingJacket* component
- Connection of a *Sliders* at input Q

Overwriting state variables

Using the operating window of the component, the operator can set the following state variables to any permitted value during simulation:

- Total mass m
- Temperature T
- Total mass fraction z
- Level L

To do so, the setpoint is initially entered in the corresponding digital input *SetValues* and applied by clicking the "Set" button. The state "Total mass" can hereby be specified with the mass value as well as with the level. By influencing the state variables, you can, for example, speed up heating processes or fill processes. The *Flash* can also be emptied, for example, to jump back in a sequencer.

Borderline case "empty flash"

Inflows to and outflows from the flash are generally throttled at the tank nozzle. During normal operation, this throttling is not immediately recognizable for the user, because throttling should have very little impact on the simulation. The outflow is severely throttled for an empty tank. The flow coefficient is thereby set to the value for maximum throttling of all connectors through which water or steam flows out. These flow coefficients are calculated internally based on the tank volume and cannot be manipulated by the modeler. The tank is considered empty when the liquid amount *mL/ρ* is less than a specified minimum percentage tank level *MinLiquidVolumePercent*:

$m_L < MinLiquidVolumePercent \cdot Volume \cdot \rho$

Balancing of the states is stopped for an empty flash, which means changes to the total mass, component mass and specific enthalpy are discarded. The "empty" state continues until there is a sufficient increase in the amount of liquid. A corresponding indicator for an empty tank is shown in the component symbol (see figure).

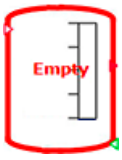


Figure 8-104 Empty flash indicator in the symbol

Borderline case "full flash"

The flash is considered to be full when its amount of liquid reaches the maximum possible percentage value:

$m_L > (V - MinVaporVolumePercent \cdot Volume) \cdot \rho$

Balancing of the states is stopped for a full flash, which means changes to the total mass, component mass and specific enthalpy are discarded. The "full" state continues until there is a sufficient decrease in the amount of liquid. A corresponding indicator for a full flash is shown in the component symbol (see figure).

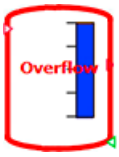


Figure 8-105 Full flash indicator in the symbol

Parameter

Parameter name	Description	Unit	Default value
Volume	Volume *V* of the tank; can be adjusted online	m^3	1.0
HeightOrLength	Height of the vertical or length or the horizontal tank; modifiable online	m	1.0
NbrOfConnectors	Number N of connectors	–	3

Parameter name	Description	Unit	Default value
NbrOfMeasurements	Number N of measuring points	–	1
MeasurementHeight	Height of the measuring points above the tank bottom	m	0.0
ConnectorHeight	Height of the pipe connectors above the tank bottom	m	0.0
Position	Orientation of the tank (vertical or horizontal)	–	vertically

Additional parameters

Parameter name	Description	Unit	Default value
InitLevelPercent	Initialization level	%	50.0
InitTemperature	Initialization temperature	°C	80.0
Initz	Initialization composition low boiler	kg/kg	0.5
InitEvaporationPart	Initialization mass fraction of the low boiler	%	0.0
InitcpLiquid	Heating capacity of the liquid for initialization	kJ/kg/K	4.18
InitcpVapor	Heating capacity of the gas phase for initialization	kJ/kg/K	1.0
InitDensityLiquid	Liquid density in the tank for initialization	kg/m³	997.337
InitRS	Specific gas constant for initialization	kJ/kg/K	0.287
geoHeight	Geodetic height of the tank	m	0.0
T_LB	Boiling temperature low boiler	°C	60
T_HB	Boiling temperature high boiler	°C	100
S	Boiling lens spread	–	10
hv_HB	High-boiler enthalpy of evaporation	kJ/kg/K	2300
hc_LB	Low-boiler enthalpy of evaporation	kJ/kg/K	1800
T2_HB	Boiling temperature high boiler for *p2_HB*	°C	120
P2_HB	Pressure for boiling temperature *T2_HB*	Bar	2.0
MinVaporVolumePercent	Percentage of tank volume that cannot be filled with liquid when filling the tank	%	5
MinLiquidVolumePercent	Percentage of tank volume that remains in the tank when draining the tank	%	0.5

Operating window

Variable	Symbol	Unit	Can variable be changed?
Level	L	m	Yes
Temperature	T	°C	Yes
Total mass	m	kg	Yes
Pressure	p	bar	No
Low-boiler mass fraction	z	%	Yes

Variable	Symbol	Unit	Can variable be changed?
Mass fraction low boiler liquid phase	x	%	No
Mass fraction low boiler vapor phase	y	%	No
Vapor fraction	vf	%	No

HeatingJacket – Heating jacket

Symbol

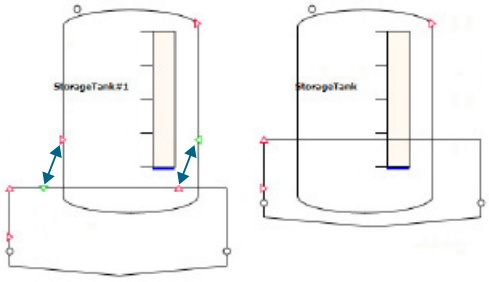


Figure 8-106 Symbol of component type HeatingJacket in connection with component StorageTankLiquid, representation of the connectors to be connected

Function

The *HeatingJacket* component is optimized for use in combination with a *StorageTankLiquid*. It represents an add-on component for the *StorageTankLiquid* but can also be used in combination with the pressure strainer or on its own.

Use in combination with a StorageTankLiquid

When the *HeatingJacket* component is used in connection with a *StorageTankLiquid*, make sure that the signal exchange between the two components takes place over two connectors. These are:

- Heat flow Q (output from the *HeatingJacket*, input for the *StorageTankLiquid*)
- Measurement point connector M (output from the *StorageTankLiquid*, input for the *HeatingJacket*)
  The measurement point connector used here must be located on the tank at tank height "zero" or below the anticipated liquid level (MeasurementHeightparameter)

The size of the two components and the position of the named connectors must be preset in such a way that the two components can be put on top of each other – as shown on the left in the figure above – so that the corresponding connections are closed automatically. In this case the connectors disappear as shown on the right in the figure above. Alternatively, you can draw the connections using connecting lines.

The *HeatingJacket* component must be linked to a flow network as a branch element.

A (compensating) heat flow Q results due to the temperature difference that arises in the *HeatingJacket* and in the *StorageTankLiquid*.

This heat flow is calculated in a simplified manner using the following equation:

$$\dot{Q} = k\ A\ \Delta T$$

k	Heat transfer coefficient
Q	Heat transfer area m^2
ΔT	Temperature difference between heating/cooling medium and tank temperature

Influence of an agitator on the heat transfer

You can represent the influence of an agitator on the heat transfer using the *AgitatorInfluence* parameter. With this parameter, the calculated heat transfer Q is scaled as a function of the agitator speed. 100% hereby corresponds to the full heat transfer and 0% to no heat transfer.

The default setting for the *AgitatorInfluence* parameter is 100% across the entire speed range of the agitator, which means always full heat transfer, no influence of the agitator on the heat transfer.

You can use the *AgitatorSpeed* input to link the agitator speed to the *HeatingJacket* component.

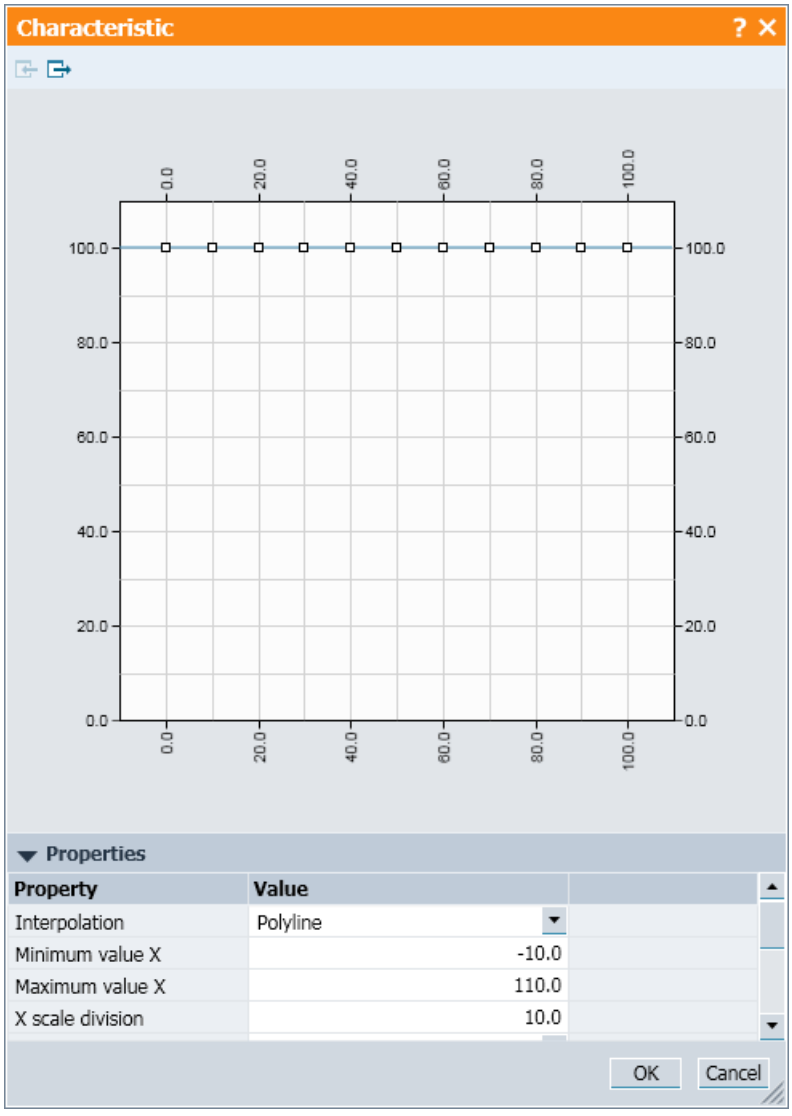


Figure 8-107 Parameter AgitatorInfluence of component type HeatingJacket

Influence of the level on the heat transfer

You can represent the influence of the tank level on the heat transfer using the *LevelInfluence* parameter. With this parameter, the calculated heat transfer is scaled as a function of the tank level. 100% hereby corresponds to the full heat transfer and 0% to no heat transfer.

The default setting for this parameter is 100% across the entire fill level range of the tank, which means no influence of the level on the heat transfer.

You can use the *TankLevel* input to link the tank level to the *HeatingJacket* component. Use the tank level in percent for this.

The influence of the agitator and influence of the level are combined with one another by multiplication.

Parameters

Parameter name	Description	Unit	Default value
Kvs	Valve characteristic value k_{VS} with $k_{VS} \geq k_{V0} + 10^{-6}$ m^3/h	m^3/h	10.0
NbrOfMeasurements	Number N of measuring points	–	1
AgitatorInfluence	Influence of an agitator on the heat transfer, in percent, as a function of the agitator speed	–	
LevelInfluence	Influence of the tank level on the heat transfer, in percent, as a function of the tank level	–	

Additional parameters

Parameter name	Description	Unit	Default value
Volume	Volume of the heating jacket	m^3	0.1
Area	Heat transfer area	m^2	10.0
k	Heat transfer coefficient [$kW/(m^2 \cdot K)$]	–	1.0
InitialTemperature	Initialization temperature	°C	20.0
MinEnthalpy	Lowest enthalpy	kJ/kg	–300.0
MaxEnthalpy	Highest enthalpy	kJ/kg	3000.0

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Temperature	T	°C
Current flow	–	kg/s

The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Volume	–	m^3
kvs	h	m^3/h
Heat transfer area	–	m^2

LiquidLiquidTank

Symbol

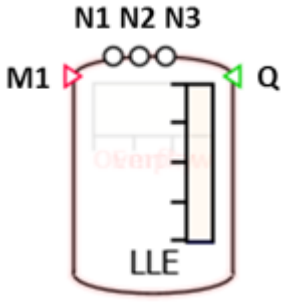


Figure 8-108 LiquidLiquidTank

Function

The component type *LiquidLiquidTank* calculates the mixing and storing of liquid media that form two phases. The incoming and outgoing substance flows are balanced with respect to mass and enthalpy. Temperature control of the mass accumulated in the tanks is possible using a direct input or various additional components.

If there are gas connectors, these are balanced separately and form a gas phase above the liquid in the tank. The mass balanced in the tank is a mixture of a maximum of two liquid components. The concentration of these components in the inflows can be specified at each input.

The light phase (*LightPhase*) and the heavy phase (*HeavyPhase*) separate immediately in the tank. The heavy phase settles at the bottom. The light phase can also have a residual solubility for the heavy phase (*SolubilityHP*), or the heavy phase can have a residual solubility for the light phase (*SolubilityLP*).

Interconnection

The *N1...Nn* connectors are the connection points for the material flows, which means for interconnection of pipes. The number of these connection points depends on the system to be simulated and is specified with the *NbrOfConnectors* parameter. All connectors of the input vector *N1...Nn* are initially equivalent and can be used for all connector types (water/steam, ideal gas, liquid). You can specify the fraction of the light phase in the inflowing medium in mass percent for each of the connectors *N1..Nn* using the input vector *LightPhaseFraction* Inputs for gas connectors are ignored and are disregarded in the balance calculation of the liquid phases. Depending on the level of the phases and the installation height of the nozzle, all outgoing mass flows exit as light phase or heavy phase, whereby the residual solubility is taken into account.

The medium in the adjoining flow networks is defined using the *NetParam* component. All defined inputs in the input vector *N1...Nn* must be interconnected. None of these inputs may remain open during simulation because this would create open flow networks. This would, in turn, result in an error message during code generation.

For the water/steam connector type, the incoming medium is mixed internally in the tank with a liquid medium. The inflowing mass and enthalpy are taken into account but the model does

not form a corresponding phase equilibrium. Incoming steam does not contribute to pressure build-up in the gas phase. Water is treated in the *LiquidLiquidTank* as a liquid medium, and the specifications for *LightPhaseFraction* are taken into account.

Parameter *ConnectorHeight*

For each of the connection points *N1*...*Nn*, the installation height of the nozzle on the tank (*ConnectorHeight* parameter) is needed. The information is entered in meters. The default setting is 0 m, which means at the bottom of the tank. Negative values are permitted for simulation of connectors at a lower position. If a gas connector is below the level of the tank, neither liquid nor gas can flow out of the container through that connection.

Balancing

The liquid phase is balanced with respect to the incoming and outgoing mass flows and enthalpies. In addition, a mass balance of the light phase is carried out.

If at least one gas connector is connected, a corresponding gas pressure is also calculated. In this case, a closed and thus pressure-tight storage tank is assumed. The gas pressure is changed by adding or removing gas and/or changing the gas volume (e.g. by filling the tank). The calculated gas pressure also affects the liquid connectors. The status of the gas balancing is displayed in the symbol with an orange background in the level display. The gas pressure is then calculated according to the state equation of ideal gases:

The gas temperature T_{Gas} is calculated from the balanced gas enthalpy h_{Gas}. If the gas temperature and liquid temperature are different, a corresponding temperature compensation occurs during simulation.

If a gas connector is not connected, the gas balancing is omitted In this case, the gas pressure is replaced with the atmospheric pressure. In this case it is assumed that the tank is open to the surrounding environment.

Representation of the balancing in the storage tank with and without gas fraction:

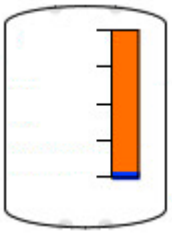


Figure 8-109 Balancing with gas fraction

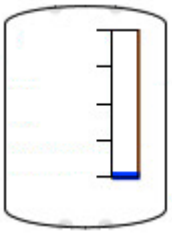


Figure 8-110 Balancing as open tank

Orientation

The *LiquidLiquidTank* component can be operated either as a vertical tank or a horizontal tank. The orientation can be set with the *Position* parameter. The tank is then simulated as a vertical or horizontal cylinder. The *HeightOrLength* parameter indicates the height of the tank for the vertical tank and the length of the tank for the horizontal tank.

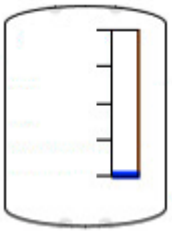


Figure 8-111 Vertical tank

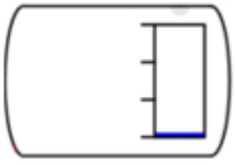


Figure 8-112 Horizontal tank

Heating/cooling of the tank

The *LiquidLiquidTank* can be heated or cooled using the following options:

- Combination with *HeatingJacket* component
- Combination with *ElectricalHeatingJacket* component
- Via specification of a heat flow at input *Q*

To simplify matters, the input heat flow is used exclusively for heating the liquid phase. A heat exchange takes place at the boundary interface between the liquid and gas phases.

Overwriting state variables

By using the operating window of the component, you can set these two state variables during simulation to any permissible value:

- Accumulated mass
- Temperature

To do so, the setpoint is initially entered in the corresponding digital input *SetValues* and applied by clicking the "Set" button. The "accumulated mass" state can be specified using the value of the mass as well as using the level (*Level*). By influencing the state variables, you can, for example, speed up heating or filling operations. The tank can also be emptied, for example, to jump back in a sequencer.

Borderline case "empty tank"

Inflows into the tank and outflows from the tank are generally throttled at the nozzle of the tank. During normal operation, this throttling is not immediately recognizable for the user, because throttling should have very little impact on the simulation. For an empty tank, the outflow becomes strongly throttled. The flow coefficient is therefore set to the value for maximum throttling for all connectors via which liquid flows out. These flow coefficients are internally calculated as a function of the tank volume and cannot be influenced by the modeler. The tank is considered empty when the liquid quantity mL/ρ is less than a specified minimum tank level in percent *MinLiquidVolumePercent*:

The balancing of the states is stopped for an empty tank. That is, changes of mass, partial masses and specific enthalpy are discarded. The "empty" state continues until there is a sufficient increase in the liquid quantity. That is, the result of inflowing quantity minus outflowing quantity is positive. A corresponding indicator for an empty tank is shown in the component symbol.

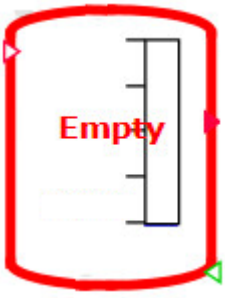


Figure 8-113 Empty tank

Borderline case "full tank"

The tank is considered to be full when its liquid quantity reaches the maximum possible percentage value and mV/ρ is thus less than a given minimum tank level in percent *MinGasVolumePercent*.

The balancing of the states is stopped for a full tank. That is, changes of accumulated mass, partial mass and specific enthalpy are discarded. The "full" state continues until there is a sufficient decrease in the liquid quantity. That is, the result of inflowing quantity minus outflowing quantity is negative. A corresponding indication of a full tank is shown in the component symbol.

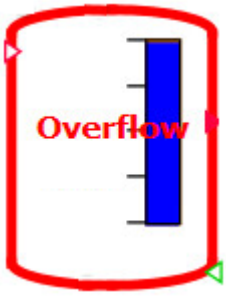


Figure 8-114 Full tank

Parameters

Parameter name	Description	Unit	Default value
Volume	Volume V of the tank	m^3	1.0
HeightOrLength	Height of the vertical or length or the horizontal tank	m	1.0
NbrOfConnectors	Number N of connectors	-	1
NbrOfMeasurements	Number N of measuring points	-	1
MeasurementHeight	Height of the process tags above the tank bottom	m	0.0
ConnectorHeight	Height of the pipe connectors above the tank bottom	m	0.0
Position	Orientation of the tank (vertical or horizontal)	-	vertical

Additional parameters

Parameter name	Description	Unit	Default value
PressureOutside	Ambient pressure	bar	1.0
LevelInit	Initialization level	%	3.0
TemperatureInit	Initialization temperature	°C	20.0
PressureInit	Initialization pressure	bar	1.0
InitLightPhase	Initialization mass fraction of light phase	%	20.0
DensityLP	Density of light phase	kg/m^3	800.0
DensityHP	Density of heavy phase	kg/m^3	1000.0
SolubilityLP	Residual solubility of the light phase in the heavy phase	kg/kg %	0.0
SolubilityHP	Residual solubility of the heavy phase in the light phase	kg/kg %	0.0
MinGasVolumePercent	Percentage of tank volume that cannot be filled with liquid when filling the tank	%	5
MinLiquidVolumePercent	Percentage of tank volume that remains in the tank when draining the tank	%	0.5
geoHeight	Geodetic height of the tank	m	0.0

Operating window

Variable	Symbol	Unit	Can variable be changed?
Level	L	m	Yes
Level of heavy phase	L_{HP}	m	No
Temperature of liquid	T_L	°C	Yes
Temperature of gas	T_G	°C	No
Mass	M	kg	Yes

Pressure at tank bottom	p_B	bar	No
Gas pressure	p_G	bar	No
Mass fraction of light phase	w	%	Yes

StorageTankLiquid – Storage tank for liquid media

Symbol

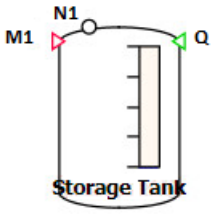


Figure 8-115 Symbol of component type StorageTankLiquid

Function

The component type *StorageTankLiquid* is used for mixing and storage of liquid media. The incoming and outgoing mass flows are hereby automatically balanced regarding mass and enthalpy.

The temperature of the substance stored in the tank can be adjusted by means of a direct input or different additional components.

If there are gas connectors, these are balanced separately and form a gas phase above the liquid in the tank.

The mass balanced in the tank is a mixture of up to two liquid components. The concentration of these components in the inflows can be specified at each input.

A thermal separation of the two components is possible during the simulation. These components are therefore referred to in the following as low boiler and high boiler components.

Other physical processes, such as the mixing of concentrations, are also possible.

Circuit

The *N1 ..Nn* connectors are the connection points for the material flows, which means for the connection of pipes.

The number of these connection points depends on the system to be simulated and is specified with the *NbrOfConnectors* parameter. All connectors of the input vector *N1..Nn* are initially equal and can be used for all connection types (water/steam, ideal gas, liquid).

Possible connection types include:

- Liquid inflows of different chemical substances
- Outflows of substances stored in the tank
- Inflow/discharge of cleaning agents

- Inflow/discharge of detergents
- Gas inflow and outflow
- Air extraction

The medium in the neighboring flow networks is defined over the *NetParam* component.

You must connect all defined inputs in the input vector *N1 ..Nn*. These inputs must not remain open during simulation because this will create open flow networks. This would, in turn, result in an error message during code generation.

For the water/steam connector type, the incoming medium is mixed internally in the tank with a liquid medium. The inflowing mass and enthalpy are taken into account but the model does not form a corresponding phase equilibrium. Incoming steam does not contribute to pressure build-up in the gas phase. Water is handled in the mixture like a liquid medium.

Due to the lack of water/steam properties, steam sterilization with a water/steam flow network connector is currently not possible.

Parameter ConnectorHeight

For each of the connection points *N1..Nn*, the installation height of the nozzle on the tank (*ConnectorHeight* parameter) is needed. The information is entered in meters. The default is 0 m, which means at the bottom of the tank. Negative values are permitted for simulation of connectors at a lower position. If a gas connection is below the level of the tank, neither liquid nor gas can flow out of the container through that connection.

Input vector EvapFraction

You can use the input vector *EvapFraction* to specify the fraction of the low-boiler in the inflowing medium, in percent, for each of the connectors *N1..Nn*. Connectors without low-boiling mixture component (e.g. detergents) are given the value "0" (default). Gas connectors are also given the value "0".

The specified values only apply to incoming flows. All outgoing flows exit with the fraction of low boiler in the stored mass *Rate*. This fraction is continuously calculated based on the incoming and outgoing mass flows and their *EvapFraction* parameters.

In the case of outgoing mass flows, any *EvapFraction* that may have been entered is not evaluated.

Balance calculation of the liquid phase

The liquid phase is balanced with respect to the incoming and outgoing mass flows and enthalpies. Mass balancing is also carried out for the low boiler in the mixture.

$$\frac{dm_{Liquid}}{dt} = \sum_{i=1}^{n} \dot{m}_{Liquid,i} - \dot{m}_{Evap}$$

$$\frac{dm_{EvapFraction}}{dt} = \sum_{i=1}^{n} \dot{m}_{EvapFraction,i} - \dot{m}_{Evap}$$

$$\frac{dh_{Liquid}}{dt} = \frac{1}{m_{Liquid}} \left(\sum_{i}^{n} \dot{m}_{Liquid,i}\, h_{Liquid,i} + \sum \dot{Q} - \dot{m}_{Evap} \left(h' + \Delta H_V \right) \right)$$

$$T_{Liquid} = \frac{h_{Liquid}}{c_P}$$

Gas balance calculation

If at least one gas connector is connected, a corresponding gas pressure is also calculated. In this case, a closed and thus pressure-tight storage tank is assumed. The gas pressure is changed by adding or removing gas and/or changing the gas volume (e.g. by filling the tank). The calculated gas pressure also affects the liquid connectors. The status of the gas balancing is obtained from an orange background in the level display.

The incoming and outgoing mass flows and enthalpies are balanced within the vapor space.

$$\frac{dm_{Gas}}{dt} = \sum_{i}^{n} \dot{m}_{Gas,i} - \dot{m}_{Evap}$$

$$\frac{dh_{Gas}}{dt} = \frac{1}{m_{Gas}} \left(\sum_{i}^{n} \dot{m}_{Gas,i}\, h_{Gas,i} + \dot{m}_{Evap} \left(h' + \Delta H_V \right) \right)$$

The gas pressure is then calculated according to the state equation of ideal gases:

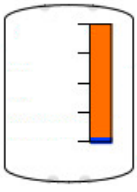
$$p_{Gas} = \frac{m_{Gas}\, R_S\, T_{Gas}}{V_{Gas}}$$

The gas temperature T_{Gas} is calculated from the balanced gas enthalpy h_{Gas}. If the gas temperature and liquid temperature are different, a corresponding temperature compensation occurs during simulation.

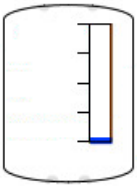
If a gas connector is not connected, the gas balance calculation does not take place. In this case, the gas pressure is replaced with the atmospheric pressure. In this case we assume that the tank is open to the surrounding environment.

Representation of the balancing in the storage tank with and without gas fraction:

Balancing with gas fraction



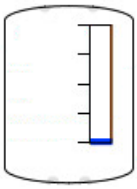
Balancing as open tank



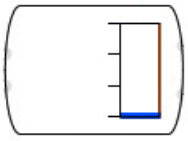
Orientation

The *StorageTankLiquid* component can be operated as either a vertical or a horizontal tank. The orientation can be set with the *Position* parameter. The tank is then simulated as a vertical or horizontal cylinder. The *HeightOrLength* parameter specifies the height of vertical tanks and the length of horizontal tanks.

Vertical tank



Horizontal tank



Evaporation

If the stored liquid contains a low boiler, it is possible to evaporate it. To do this, the liquid mixture must be heated to a temperature above the boiling point corresponding to the current pressure. The steam pressure curve needed for this is configured using the *Boiling* parameter (vapor pressure curve of the low boiler $p_S = f\ (T_S)$ in [bar]).

In addition, the evaporization enthalpy of the low boiler must be specified as a characteristic curve ($\Delta H_V = f(T_S)$ in [kJ/kg]) using the *Vaporization* parameter.

The respective characteristic curves for water are used by default. They can be edited within the charts or imported from a text file.

Examples of these characteristic curves are shown in the figures below:

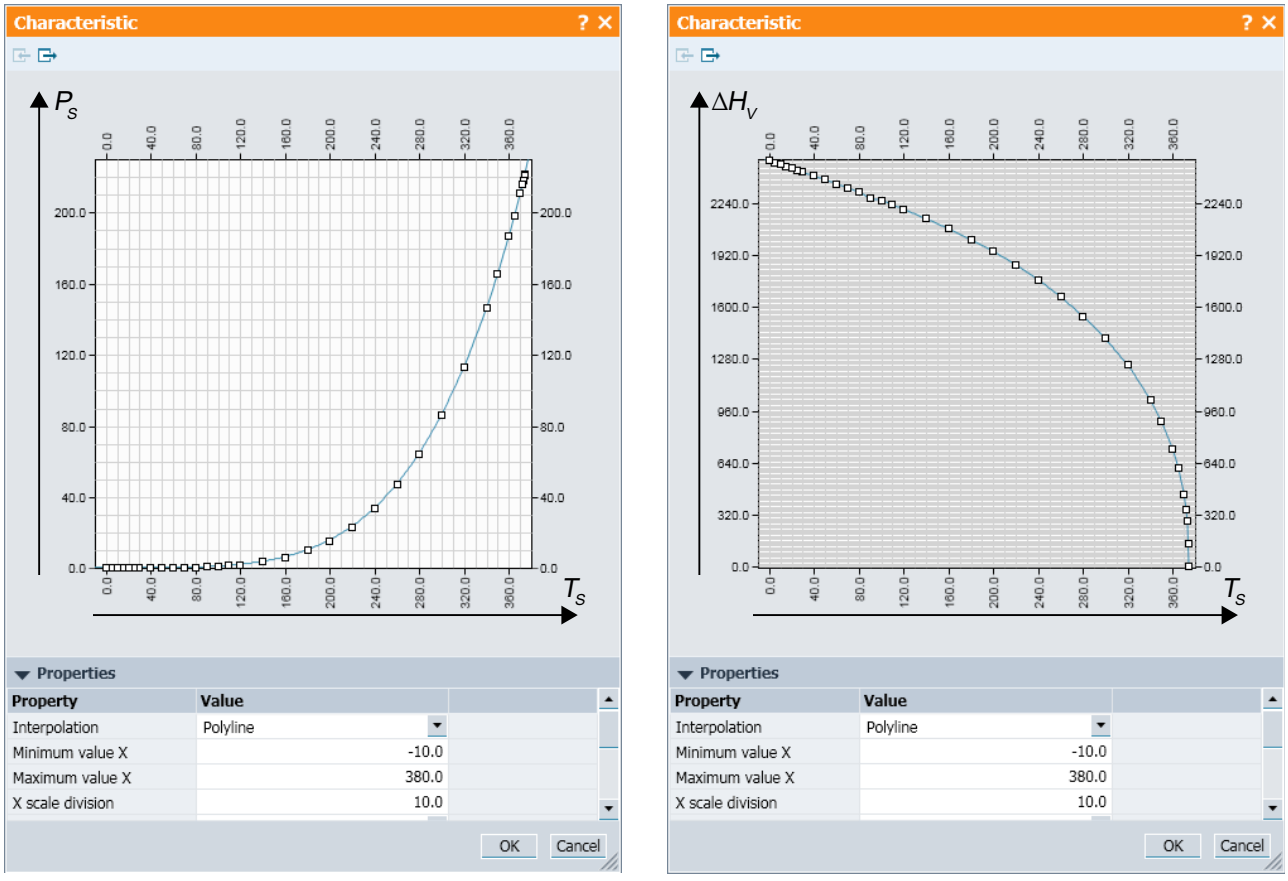


Figure 8-116 Example for vaporization enthalpy and vapor pressure curve

When the vaporization condition $T > T_S$ (p_S) has been reached, the low boiler gradually transforms to the gas phase and is added to the gas balance. No provision is made for condensation in the tank.

Heating / cooling of the storage tank

The *StorageTankLiquid* can be heated or cooled using the following options:

- Combination with *HeatingJacket* component
- Combination with *ElectricalHeatingJacket* component
- Connection of a slider at the *Q* input

To simplify matters, the supplied heat flow is only used to heat the liquid phase.

Pressure drop at the nozzles

The flow coefficient *KV* of the pressure loss to the tank nozzle is calculated depending on the pressure loss coefficient ζ, the container size *V*,a typical dwell time of 3 minutes and a typical flow velocity in the nozzles of 2.5 m/s.

$$KV = 60\frac{V}{\tau\, v}\sqrt{\frac{200}{\zeta}}$$

The following applies:

$$KV = \dot{Q}\sqrt{\frac{\rho}{\Delta p}\frac{1}{1000}}$$

$$\Delta p' = \frac{1}{2}v^2\zeta\rho$$

$$\dot{V} = \frac{V}{\tau\, 60}$$

With:

Δρ:	Pressure drop in [bar]
Δρ':	Pressure drop in [Pa]
Q:	Volume flow in [m³/h]
V:	Volume flow in [m³/s]
ρ:	Density in [kg/m³]
KV:	Flow coefficient in [m³/h]
τ:	Dwell time [min]
v:	Flow velocity in [m/s]
ζ:	Pressure drop coefficient [-]

By adjusting the pressure loss coefficient (parameter Zeta ζ) the pressure drop of all nozzles can be varied.

Overwriting state variables

By using the operating window of the component, you can set these two state variables during simulation to any permissible value:

- Stored mass
- Temperature

For this, you must first enter the setpoint in the corresponding digital input *SetValues* and apply it with the "Set" button. The state "stored mass" can be specified using the value of the mass as well as using the level (*Level*).

By manipulating state variables in this way, you can, for example, speed up heating processes or filling processes. Likewise, you can empty the tank, for example, to return to an earlier step in a step sequence.

Borderline case "Tank empty"

Inflows into the tank and outflows from the tank are generally throttled at the nozzle of the tank. During normal operation, this throttling is not immediately recognizable for the user, because throttling should have very little impact on the simulation.

The outflow is severely throttled for an empty tank. The flow coefficient is thereby set to the value for maximum throttling of all connectors through which water or steam flows out.

These flow coefficients are calculated internally based on the tank volume and cannot be manipulated by the modeler.

The tank is considered empty when the liquid amount mL/ρ is less than a specified minimum percentage tank level *MinLiquidVolumePercent*:

$m_L < RemainderLiquidPercent \cdot Volume \cdot \rho$

The balancing of the states is stopped for an empty tank. That is, changes of mass, partial masses and specific enthalpy are discarded. The status "empty" only changes when the amount of liquid increases sufficiently, which means the difference between inflowing and outflowing quantity is positive.

A corresponding indicator for an empty tank is shown in the component symbol (see figure).

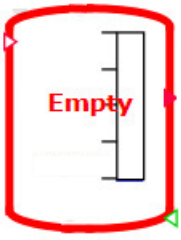


Figure 8-117 Display of an empty tank in the symbol

Borderline case "Tank full"

The tank is considered to be full when its amount of liquid reaches the maximum possible percentage value:

$m_L < (V - RemainderGasPercent \cdot Volume) \cdot \rho$

Balancing of the states is stopped for a full tank, which means changes to the mass, partial mass and specific enthalpy are discarded. The status "full" only changes when the amount of liquid decreases sufficiently, which means the difference between inflowing and outflowing quantity is negative.

A corresponding note is shown in the symbol of a component for a full tank (see figure).

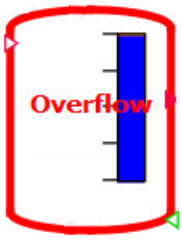


Figure 8-118 Display of the full tank in the symbol

Parameters

Parameter name	Description	Unit	Default value
Volume	Volume *V* of the tank; can be adjusted online	m^3	1.0
HeightOrLength	Height of the vertical or length or the horizontal tank; modifiable online	m	1.0
NbrOfConnectors	Number N of connectors	–	1
NbrOfMeasurements	Number N of measuring points	–	1
MeasurementHeight [Nr]	Height of the measuring points above the tank bottom	m	0.0
ConnectorHeight [Nr]	Height of the pipe connectors above the tank bottom	m	0.0
Boiling	Vapor pressure curve of the low boiler: Pressure in bar as function of the temperature (°C)	–	
Vaporization	Evaporization enthalpy of the low boiler in kJ/kg as a function of temperature (°C)	–	
Position	Orientation of the tank (vertical or horizontal)	–	vertically

Additional parameters

Parameter name	Description	Unit	Default value
PressureOutside	Ambient pressure	bar	1.0
LevelInit	Initialization level	%	3.0
TemperatureInit	Initialization temperature	°C	20.0
PressureInit	Initialization pressure	bar	1.0
InitEvaporationPart	Initialization mass fraction of the low boiler	%	0.0
Density	Mixture density in the tank	kg/m^3	997.337
RemainderGasPercent	Percentage of tank volume that cannot be filled with liquid when filling the tank	%	5
RemainderLiquidPercent	Percentage of tank volume that remains in the tank when draining the tank	%	0.5
geoHeight	Geodetic height of the tank	m	0.0
zeta ζ	Pressure drop coefficient	-	0.05

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit	Variable can be changed
Level	L	m	Yes
Temperature of fluid	T_L	°C	Yes
Temperature of gas	T_G	°C	No
Mass	m	kg	Yes
Pressure at tank bottom	p_B	bar	No
Gas pressure	p_G	bar	No
Low-boiler mass fraction	w	%	Yes

See also

Geodetic height (Page 618)

8.2.5.12 Valves

ControlValve / Valve / AngleValve

Symbol

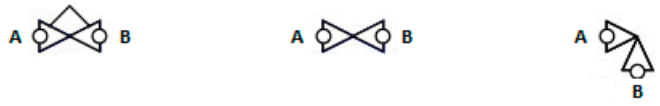


Figure 8-119 Symbols of the component types ControlValve / Valve / AngleValve

Function

The component types *ControlValve*, *Valve* and *AngleValve* are used to simulate a control valve. They can also be used for butterfly valves, plug valves or gate valves.

The listed components differ only in their symbols. The behavior of the components is identical.

Depending on the valve position, the configured characteristic curve and the valve size (k_{VS} value), the pressure drop across the valve is calculated according to the formula below.

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_{VS}^2 \, \rho \frac{\text{kg}}{\text{m}^3}} 12960 \left(\frac{\text{sec}}{\text{h}} \right)^2$$

Whereby:

$\Delta p = p_B - p_A$	the pressure drop across the valve in bar
$\dot{m}$	the flow or mass flow in kg/s
ρ	the density of the medium in kg/m³
k_{VS}	the valve characteristic value in m³/h

The reference direction is defined for the mass flow $\dot{m}$ from connector *A* to connector *B*. This means $\dot{m} > 0$ for mass flow in the reference direction. The pressure drop then has a negative value: $\Delta p < 0$. This is, however, purely a counting direction, which means the direction in which the valve is installed is irrelevant. The flow direction can also be reversed during the simulation.

The control of these components through the control technology is not included in the component types. These are mapped in the actuator/sensor level. The interface between the drive level of a valve and the component types *ControlValve*, *Valve* and *AngleValve* in the model level is the valve position.

This position value is limited to the range 0 ≤ Position ≤ 100%. The valve position is mapped to the valve characteristic value k_I using the valve characteristic.

You can select the following valve characteristic curves:

- Linear characteristic
- Square-law characteristic
- Equal-percentage characteristic
- Graphically configured characteristic (polygon)

Parameters

Parameter name	Description	Unit	Default value
Characteristic	Selection of the valve characteristic	–	*Linear*
Kvs	Valve characteristic value k_{VS} with $k_{VS} \geq k_{V0} + 10^{-6}$ m³/h	m³/h	10.0
OpeningCharacteristic	Specification of the valve characteristic as characteristic curve. This parameter is only used if the value *Polygon* has been selected for the *Characteristic* parameter. The opening in percentage is hereby specified as a function of the stroke in percentage.		

Additional parameters

Parameter name	Description	Unit	Default value
Kv0	Valve characteristic value k_{V0} with $k_{V0} \geq 10^{-6}$ m³/h	m³/h	0.000001

Operating window

You can switch between automated control (control using the actuator/sensor level) and manual control in the operating window. The default is always automated control.

The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Pressure difference	Δp	bar
Mass flow	$\dot{m}$	kg/s

RuptureDisc – Rupture disc

Symbol



Figure 8-120 Symbol of component type RuptureDisc

Function

The component type *RuptureDisc* simulates the functionality of a rupture disc. Depending on a configurable response pressure, a diaphragm is irrevocably damaged or, from a simulation point of view, a valve is irreversibly opened.

However, you can manually close the rupture disk again in the simulation model after it has ruptured with the "Reset" button in the operating window. This function is needed in order to test the rupturing of the rupture disc several times in succession without having to reset the simulation.

You can make use of this function only as a manual function via the operating window. It cannot be connected to other components or the control technology.

For simulation purposes, the rupture disc is simply considered to be a closed valve that opens permanently when the response pressure is exceeded. The parameter assignment is therefore the same as for a valve.

Parameters

Parameter name	Description	Unit	Default value
Kvs	Valve characteristic value k_{VS} with $k_{VS} \geq k_{V0} + 10^{-6}$ m³/h	m³/h	100.0
OpeningPressure	Response pressure of the rupture disk	bar	10.0

Additional parameters

Parameter name	Description	Unit	Default value
Kv0	Valve characteristic value k_{v0} with $k_{v0} \geq 10^{-6}$ m³/h	m³/h	0.000001

Operating window

You can reset the rupture disk in the operating window.

The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Pressure difference	Δp	bar
Mass flow	$\dot{m}$	kg/s

During the simulation, the operating states of the *RuptureDisc* component are visualized in the symbol:

Closed rupture disc



Destroyed rupture disc



Connection example:

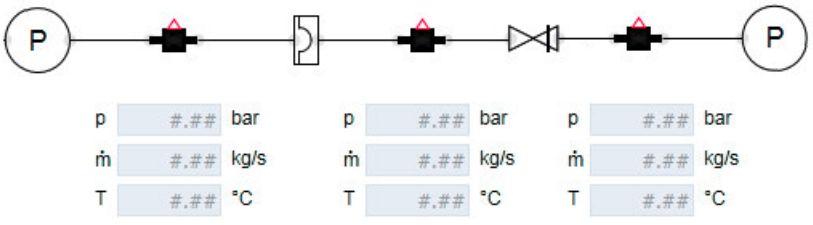


Figure 8-121 Possible connection of component type RuptureDisc

In this connection example, a rupture disc is connected upstream from a safety valve. It is important in this case that the k_{v0} value of the downstream safety valve is significantly greater than the k_{v0} value of the rupture disc.

Example: $k_{V0rupture_disc}$ = 0.000001
$k_{V0safety_valve}$ = 0.01

If these two values were set to the same value, in the case of a closed valve and rupture disc, the solution algorithm of the flow network would cause an average pressure calculated from the pressure value upstream of the rupture disk and the pressure downstream of the safety valve to arise between the rupture disc and the safety valve.

You must also select a k_{vs} value of the rupture disc that is much higher than the k_{vs} value of the safety valve.

SafetyValve / AngleSafetyValve

Symbol

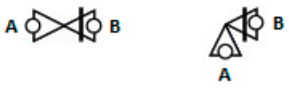


Figure 8-122 Symbols of component types SafetyValve / AngleSafetyValve

Function

The components *SafetyValve* / *AngleSafetyValve* are used as a safety valve. For a configurable pressure difference between the connectors *A* and *B*, the valve is also opened in a configurable time. The valve remains open as long as the pressure difference exists. The valve is closed automatically as soon as the pressure difference has dropped below the configured response pressure minus the configured hysteresis.

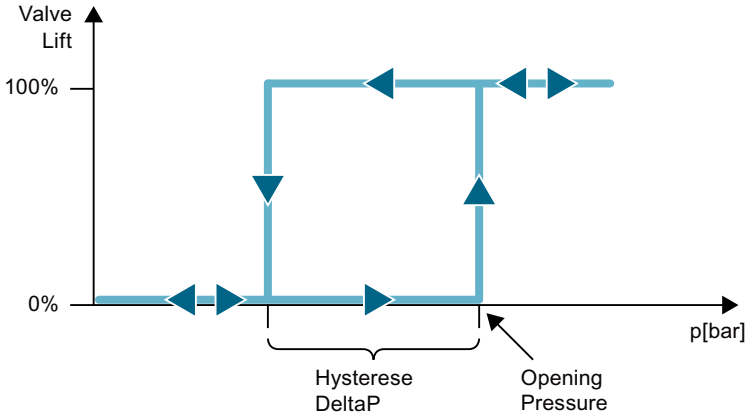


Figure 8-123 Opening and closing behavior of the safety valve

The model of the safety valve is designed with protection against polarity reversal; flow through the valve in both directions is permitted. For calculation of the pressure difference, the side on which the higher pressure exists is immaterial. This means that the safety valve cannot be inadvertently inserted in the wrong direction in the model (e.g. as the result of an automatic generation).

Note, however, that a preferred direction of flow from *A* to *B* is defined within the valve. If the flow takes place in this direction, a positive mass flow is displayed in the operating window. If the flow is in the opposite direction (i.e. the valve is mirrored, for example), this mass flow is negative. The pressure drop Δp is also negative or positive depending on the flow direction.

Parameters

Parameter name	Description	Unit	Default value
Characteristic	Selection of the valve characteristic Selection of the valve characteristic is the same as for control valves. Hysteresis is achieved with the *HystereseDeltaP* parameter.	–	*Linear*
Kvs	Valve characteristic value k_{VS} with $k_{VS} \geq k_{V0} + 10^{-6}$ m³/h	m³/h	10.0
OpeningPressure	Response pressure (pressure difference across the valve)	bar	10.0
OpeningTime	Time until valve is completely open	s	0.0
OpeningCharacteristic	Specification of the valve characteristic as characteristic curve. This parameter is only used if the value *Polygon* has been selected for the *Characteristic* parameter. The opening in percentage is hereby specified as a function of the stroke in percentage.	–	
HystereseDeltaP	Difference between response pressure and pressure drop.	bar	1.0

Additional parameters

Parameter name	Description	Unit	Default value
Kv0	Valve characteristic value k_{V0} with $k_{V0} \geq 10^{-6}$ m³/h	m³/h	0.000001

During the simulation, the operating states of the component are visualized in the symbol.

To do this, the valve position is displayed as shown in the figure below.

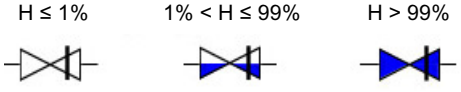


Figure 8-124 Display of the valve position in the SafetyValve symbol

All parameters can be changed online.

Operating window

The safety valve can also be opened or blocked manually in the operating window using the "local control" function. This gives you the ability to easily simulate a malfunction of the safety valve.

For manual control, you can specify the valve position using a slider. In this case the function is that of a manual valve. The dependency on the pressure drop no longer exists.

The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Pressure difference	Δp	bar
Mass flow	$\dot{m}$	kg/s

StopValve – Stop valve

Symbol



Figure 8-125 Symbol of component type StopValve

Function

The component type *StopValve* is used for simulation of a check valve. The pressure drop across the check valve is calculated depending on the direction of flow. The reference direction is defined for the mass flow $\dot{m}$ from connector *A* to connector *B*. This means $\dot{m} > 0$ for mass flow in the reference direction.

For a positive mass flow, the valve is simulated as a fully opened valve. For a negative mass flow, the component type represents a closed valve. In this case leakage becomes active; the valve coefficient k_V therefore corresponds to the minimum opening k_{V0}.

Parameters

Parameter name	Description	Unit	Default value
Kvs	Valve characteristic value k_{V0} with $k_{V0} \geq 10^{-6}$ m³/h	m³/h	10.0

Additional parameters

Parameter name	Description	Unit	Default value
Kv0	Valve characteristic value k_{V0} with $k_{V0} \geq 10^{-6}$ m³/h	m³/h	0.000001

To realize the check function, the valve characteristic value k_{V0} must be selected sufficiently small.

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure difference	Δp	bar
Mass flow	$\dot{m}$	kg/s

ThreeWayValve

Symbol

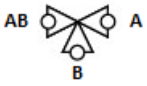


Figure 8-126 Symbol of component type ThreeWayValve

Function

The component type *ThreeWayValve* is used for simulation of a three-way valve. Depending on the valve position, the flow paths shown in the figure below can be opened.

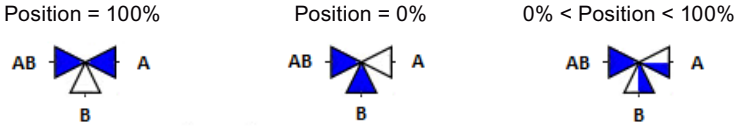


Figure 8-127 Positions of the ThreeWayValve dependent on the *Position* input

The component type has two flow branches. Flow is always present in connector *AB*. It is the connector that is shared by both valve branches. Depending on the valve position, connector *A* or connector *B* can be open (intermediate positions are possible). No flow is possible from *A* to *B* or vice versa (analog to the real three-way valve).

Depending on its insertion, the component can be used as mixing valve or as diverting valve. The anticipated flow in the valve (mixing valve or diverting valve) must already be known when the connection is made. **Insert the valve in the corresponding orientation.**

The flow direction during the simulation results from the relative pressures.

The valve position is specified at the *Position* input. This value refers to the valve branch *AB–A*. Opening of the valve branch *AB–B* results from the difference of the opening of the valve branch *AB–A* at 100%.

The pressure drop above the valve is calculated using the following formula depending on the valve position, the configured characteristic and the valve size (k_{VS} value):

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_{VS}^2 \, \rho \frac{\text{kg}}{\text{m}^3}} \, 12960 \left(\frac{\text{sec}}{\text{h}} \right)^2$$

This calculation is made for both valve branches. The following applies, for example, to the branch *AB–A*:

$\Delta p = p_{AB} - p_A$	the pressure drop across the valve in bar (*AB–A*)
$\dot{m}_{AB-A}$	the flow or mass flow from *AB* to *A* in kg/s
ρ	the density of the medium in kg/m³
$k_{V\ AB-A}$	the valve characteristic value in m³/h

The reference direction for mass flow $\dot{m}$ is from connector *AB* to connector *A* or *B*. Thus, for a mass flow in the reference direction, $\dot{m} > 0$ (**diverter valve**). The corresponding pressure drops therefore have a negative value: $\Delta p < 0$.

If the flow direction is reversed, the valve is a **mixing valve**. Here the mass flow is calculated against the reference direction: $\dot{m} > 0$ (from the connectors *A* and *B* to *AB*). The corresponding pressure drops then have a positive value: $\Delta p > 0$ (note the counting direction!).

Parameters

Parameter name	Description	Unit	Default value
Characteristic	Selection of the valve characteristic	–	*Linear*
Kvs	Valve characteristic value k_{VS} with $k_{VS} \geq k_{V0} + 10^{-6}$ m³/h	m³/h	10.0
OpeningCharacteristic	Specification of the valve characteristic as characteristic curve. This parameter is only used if the value *Polygon* has been selected for the *Characteristic* parameter. The opening in percentage is hereby specified as a function of the stroke in percentage.	–	

Additional parameters

Parameter name	Description	Unit	Default value
Kv0	Valve characteristic value k_{V0} with $k_{V0} \geq 10^{-6}$ m³/h For simulation purposes, the three-way valve represents an internal node in the flow network. Because the size of this node can be significantly different than the size of other components, you have the option of parameterizing this node here.	m³/h	0.000001
sCompressionGas	Specific compression module for low-density media (gases/ vapors, ρ < 500 kg/m³)	bar/kg	10.0
sCompressionLiquid	Specific compression module for high-density media (liquids, ρ > 500 kg/m³)	bar/kg	100.0
FactorThermalGas	Factor for enthalpy balancing for low density media (gas/steam, ρ < 500 kg/m³)	1/kg	1000.0

Parameter name	Description	Unit	Default value
FactorThermalLiquid	Factor for enthalpy balancing for high density media (liquid, ρ > 500 kg/m³)	1/kg	10.0
PressureInit	Initialization value for the pressure in the node	bar	1.0
sEnthalpyInit	Initialization value for the specific enthalpy in the node	kJ/(kgK)	100.0
TemperatureEnvironment	Ambient temperature	°C	20.0
FactorHeatExchangeEnv	Factor of proportionality for heat exchange of the medium in the node with the environment; no heat exchange occurs when the value zero is set	kW/K	0.0

Operating window

You can switch between automated control (control using the actuator/sensor level) and manual control in the operating window. The default is always automated control.

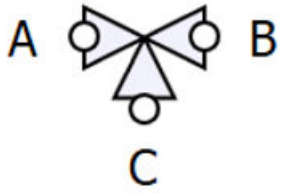
For manual control, you can specify the valve position using a slider. The slider value hereby refers to the branch *AB–A*.

The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Pressure difference	Δp_1	bar
Pressure difference	Δp_2	bar
Mass flow	$\dot{m}$	kg/s

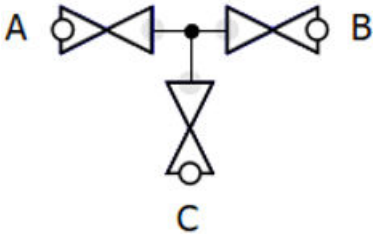
TwoWayValve - Two-way valve

Icon



Function

The TwoWayValve component type is used for simulation of a two-way valve. It corresponds in its function to an interconnection of three independent valves:



Port A, B or C, or any combination thereof, may be open depending on the control of the valve. The control is made with binary signals, nevertheless the valves open and close with a configurable running time.

The component can be used as a mixing valve or as a diverting valve.

The flow direction during the simulation results from the relative pressures.

The pressure drop across each valve is calculated using the following formula depending on the valve position, the configured characteristic curve and the valve size (k_{VS} value):

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_{VS}^2 \, \rho \frac{\text{kg}}{\text{m}^3}} \, 12960 \left(\frac{\text{sec}}{\text{h}}\right)^2$$

This calculation is performed for all valve branches. The following applies:

Δp	The pressure drop across the valve in bar
ṁ	The flow or mass flow from the connection to the internal node in kg/s
ρ	The density of the medium in kg/m³
k_V	The valve characteristic value in m³/h

The reference direction for the media flow ṁ is from the port to the internal node. For a media flow in the reference direction, ṁ > 0 applies.

Parameter

Parameter name	Description	Unit	Default value
Characteristic	Selection of the valve characteristic	-	Linear
Kvs	Valve characteristic value k_{VS} with $k_{VS} \geq k_{V0} + 10^{-6}$ m³/h	m³/h	10.0
RunningTime	Time for opening or closing the valve	s	0.5

Additional parameters

Parameter name	Description	Unit	Default value
Kv0	Valve characteristic value k_{v0} with $k_{v0} \geq 10^{-6}$ m^3/h	m^3/h	0.000001

For simulation purposes, the two-way valve represents an internal node in the flow network. Because the size of this node can be significantly different than the size of other components, you have the option of parameterizing this node here.

Parameter name	Description	Unit	Default value
sCompressionGas	Specific compression module for low-density media (gases/vapors, $\rho < 500$ kg/m^3)	bar/kg	10.0
sCompressionLiquid	Specific compression module for high-density media (liquids, $\rho > 500$ kg/m^3)	bar/kg	100.0
FactorThermalGas	Factor for enthalpy balancing for low-density media (gases/vapors, $\rho < 500$ kg/m^3)	1/kg	1000.0
FactorThermalLiquid	Factor for enthalpy balancing for high-density media (liquid, $\rho > 500$ kg/m^3)	1/kg	10.0
PressureInit	Initialization value for the pressure in the node	bar	1.0
sEnthalpyInit	Initialization value for the specific enthalpy in the node	kJ/(kgK)	100.0
TemperatureEnvironment	Ambient temperature	°C	20.0
FactorHeatExchangeEnv	Factor of proportionality for heat exchange of the medium in the node with the environment; there is no heat exchange when the value is zero	kW/K	0.0

Operating window

In the operating window, you can switch between automated control strategy (control via the actuator-sensor level) and manual control strategy (local control). The default is always automated control.

For manual control strategy, you can specify the valve positions using switches A, B and C.

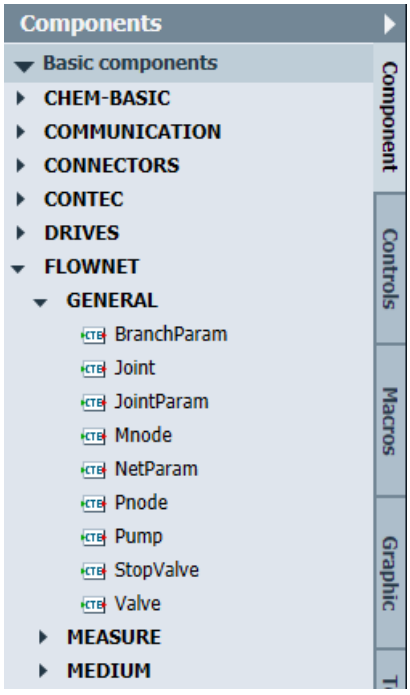
In the extended operating window, the differential pressures between the connections and the internal node are displayed.

8.2.6 Components in FLOWNET library

8.2.6.1 General components

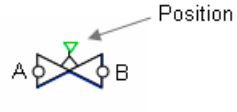
Selection of the general components

The *GENERAL* directory of the FLOWNET library includes component types that can be used in flownets with any medium.



Valve

Symbol



Function

The *Valve* component type is used for simulating a control valve. Depending on the control valve position the pressure drop over the control valve is calculated as

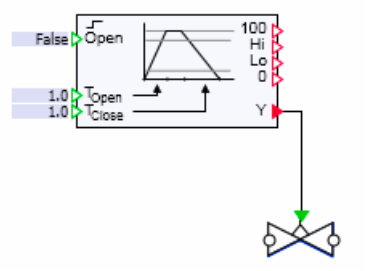
$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_V^2 \, \rho \frac{\text{kg}}{\text{m}^3}} 12960 \left(\frac{\text{sec}}{\text{h}} \right)^2$$

. The following applies:

- $\Delta p = p_B - p_A$ is the pressure drop over the valve in bar
- $\dot{m}$ is the mass flow in kg/s
- ρ is the density of the medium in kg/m³
- k_V is the flow coefficient in m³/h

The reference direction is defined for the mass flow $\dot{m}$ from connector *A* to connector *B*, which means it is $\dot{m} > 0$ for mass flow in the reference direction. The pressure drop therefore has a negative value: $\Delta p < 0$.

At the *Position* connector, the *H* position of the valve drive is given as a percentage for which the drive component types of the SIMIT basic library, for example, can be used.



This position value is limited to the range $0 \le H \le 100\%$. The control valve position is mapped using the control valve characteristic to the flow coefficient k_V. The following applies

$$\frac{k_V}{k_{V100}} = \frac{1}{S_V} + \left(1 - \frac{1}{S_V}\right) \frac{H}{100\%}$$

for a linear characteristic,

$$\frac{k_V}{k_{V100}} = \frac{1}{S_V} + \left(1 - \frac{1}{S_V}\right) \left(\frac{H}{100\%}\right)^2$$

for a quadratic characteristic,

$$\frac{k_V}{k_{V100}} = S_V^{\frac{H}{100\%} - 1}$$

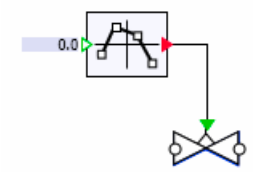
for an equal percentage characteristic.

The position ratio

$$S_V = \frac{k_{V100}}{k_{V0}}$$

is applied here as a quotient consisting of the flow coefficient *cV100* for a completely open value (*H = 100 %*) and the flow coefficient k_{V0} for a completely closed valve (*H = 0*).

Any valve characteristic curves can be created with the *Characteristic* component type from the SIMIT basic library. The characteristic component is simply placed upstream of the valve *Position* input.



The linear characteristic is then configured at the control valve so that the characteristic specified with the characteristic component is simply scaled with the factor $(k_{V100} - k_{V0}) / k_{V0}$ and offset by k_{V0}.

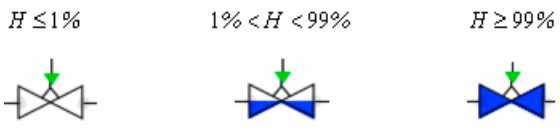
Parameters

Parameter name	Description	Unit	Default value
Characteristic	Set valve characteristic: *Linear*, *Quadratic*, *EqualPercentage*; modifiable online	–	*Linear*
Kvs	Valve characteristic value k_{V100} with $k_{V100} \geq k_{V0} + 10^{-6}$ m³/h; can be adjusted online	m³/h	360.0
Kv0	Valve characteristic value k_{V0} with $k_{V0} \geq 10^{-6}$ m³/h	m³/h	0.000001

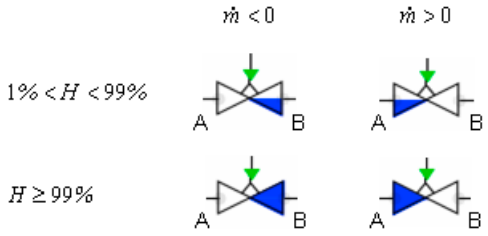
Additional parameters

Parameter name	Description	Unit	Default value
ShowFlow	Display of valve position in symbol	–	*False*
ShowFlowDirection	Display of direction of flow in symbol	–	*False*

If *ShowFlow* is set to *True* , the valve position is displayed as shown in the figure below.



If *ShowFlowDirection* is also set to *True*, the valve position and direction of flow are displayed as shown in the figure below.



Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure drop	Δp	bar
Mass flow	$\dot{m}$	kg/s

StopValve – non-return valve

Symbol



Function

The *StopValve* component type is used for simulation of a check valve. Depending on the flow direction the pressure drop over the control valve is calculated as

$$\frac{\Delta p}{\text{bar}} = \begin{cases} \dfrac{-\dot{m}^2}{k_{V100}^2\,\rho\dfrac{\text{kg}}{\text{m}^3}}\,12960\left(\dfrac{\text{sec}}{\text{h}}\right)^2 & \text{for} \quad \dot{m} > 0 \\ \dfrac{-\dot{m}^2}{k_{V0}^2\,\rho\dfrac{\text{kg}}{\text{m}^3}}\,12960\left(\dfrac{\text{sec}}{\text{h}}\right)^2 & \text{for} \quad \dot{m} < 0 \end{cases}$$

. The following applies:

- $\Delta p = p_B - p_A$ is the pressure drop over the non-return valve in bar,
- $\dot{m}$ is the mass flow in kg/s,
- p is the density of the medium in kg/m³ and
- k_V is the flow coefficient in m³/h.

The reference direction is defined for the mass flow $\dot{m}$ from connector *A* to connector *B*, which means it is $\dot{m} > 0$ for mass flow in the reference direction.

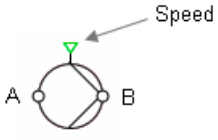
Parameters

Parameter name	Description	Unit	Default value
Kvs	Valve characteristic value k_{V100} with $k_{V100} \geq k_{V0} + 10^{-6}\ m^3/h$	m^3/h	360.0
Kv0	Valve characteristic value k_{V0} with $k_{V0} \geq 10^{-6}\ m^3/h$	m^3/h	0.000001

In order to implement the non-return function, the flow coefficient k_{V0} must be sufficiently small.

Pump – pump

Symbol



Function

The *Pump* component type calculates the pressure increase as a function of flow rate and speed according to the following formula:

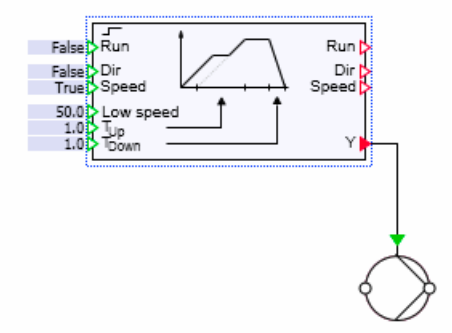
$$\Delta p = n^2 \Delta p_0 + \left(\Delta p^* - \Delta p_0\right) \frac{\dot{m}^2}{\left(\dot{m}^*\right)^2} \quad \text{for} \quad \dot{m} > 0$$

Whereby:

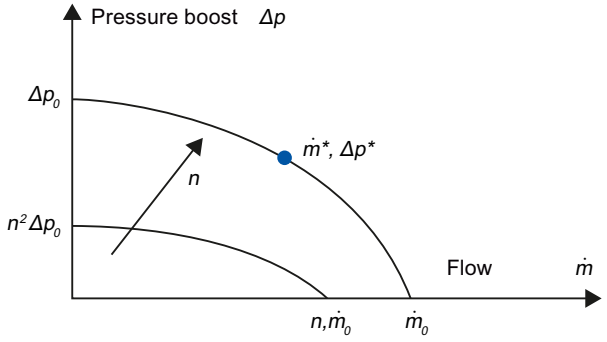
$\Delta p = p_B - p_A$ is the pressure increase in bar,
$\dot{m}$ is the mass flow in kg/s,
Δp_0 is the zero flow head in bar,
Δp^* is the nominal pressure boost in bar,
$\dot{m}^*$ is the nominal flow in kg/s and
n is the dimensionless speed value.

The reference direction is defined for the flow $\dot{m}$ from connector *A* to connector *B*, which means the following applies for a flow in reference direction $\dot{m} > 0$.

The speed *N* is specified as a percentage value at the *Speed* input. The speed value is limited to the range $0 \leq N \leq 100\%$. This means $n = N / 100\%$ and thus $0 \leq n \leq 1$. You can, for example, use the drives from the SIMIT basic library, as shown in the figure below.



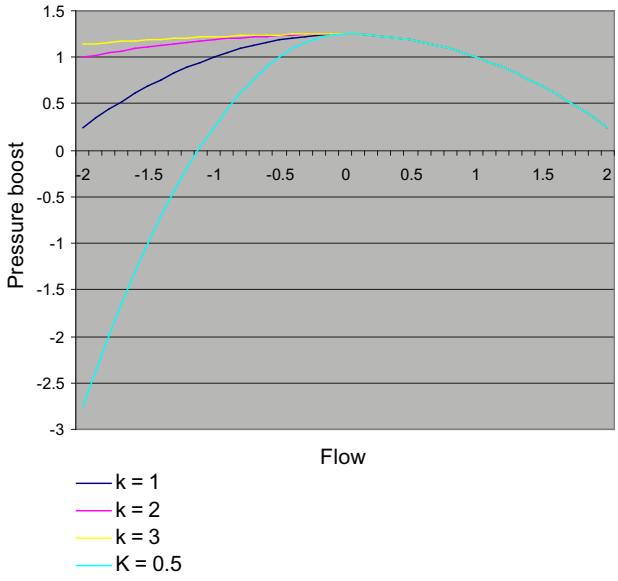
The quadratic relation between pressure boost and flow defined above is illustrated in the figure below for operation of the pump in the normal range, which means for $\dot{m} > 0$, $\Delta p > 0$.



The pump characteristic is increased steadily, according to

$$\Delta p = n^2 \Delta p_0 + \left(\Delta p^* - \Delta p_0\right) \frac{\dot{m}^2}{\left(k\dot{m}^*\right)^2} \quad \text{for} \quad \dot{m} < 0$$

when the flow is reversed ($\dot{m} < 0$). The throttling effect can be influenced by means of the flow coefficient *k*. The figure below shows the extended characteristic qualitatively for various characteristic values *k*:



It is clear that the throttling effect is greater for smaller values of the flow coefficient *k*.

To stabilize a simulation that contains a component of this type, pressure reversal ($\Delta p < 0$) is also calculated using the above equations. For switched off pumps, which means for zero speed, this results in pure throttling for the flow in pump direction as well as for the flow against pump direction:

$$\Delta p = \begin{cases} \left(\Delta p^* - \Delta p_0\right) \dfrac{\dot{m}^2}{\left(k\dot{m}^*\right)^2} & \text{for} \quad \dot{m} < 0 \\ \left(\Delta p^* - \Delta p_0\right) \dfrac{\dot{m}^2}{\left(\dot{m}^*\right)^2} & \text{for} \quad \dot{m} > 0 \end{cases}$$

Parameters

Parameter name	Description	Unit	Default value
ZeroFlowHead	Zero flow head Δp_0, $\Delta p_0 > \Delta p^*$; can be adjusted online	bar	10.0
NominalPressure	Nominal pressure Δp^*, $\Delta p^* > 0$; can be adjusted online	bar	8.0
NominalMassflow	Rated mass flow $\dot{m}^*$, $\dot{m}^* > 0$; modifiable online	kg/s	10.0

Additional parameters

Parameter name	Description	Unit	Default value
Throttling	positive flow characteristic *k (k > 0)*	–	1.0
ShowFlow	Display of operating state in symbol	–	*False*

If the additional parameter *Showflow* is set to True, the operating state in shown in the pump symbol.



Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure increase	Δp	bar
Flow rate	$\dot{m}$	kg/s

Pnode – pressure setting

Symbol



Function

The *Pnode* component type specifies the values for pressure *p* and specific enthalpy *h* at its connector A. This type of component forms a boundary for the flownet. When the flownet is represented as a graph, it corresponds to an (external) node for which the pressure and specific enthalpy are predefined.

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit	Can variable be changed?
Pressure (Default: 0)	p	bar	Yes
Specific enthalpy (Default: 100)	h	kJ/kg	Yes

The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Mass flow	$\dot{m}$	kg/s
Condensate pressure drop	Δp	bar
Density	r	kg/m³
Temperature	T	°C
Specific enthalpy	h	kJ/kg

For outflow $\dot{m} > 0$, for inflow $\dot{m} < 0$.

Mnode – mass flow setting

Symbol



Function

The *Mnode* component type specifies the values for mass flow $\dot{m}$ and specific enthalpy *h* at its connector A. This type of component forms a boundary for the flow network. When looking at the flow network as a graph, this corresponds to an inflow or outflow into an (internal) node or branch. An internal node with defined inflow or outflow is added to the flow network for each component of this type.

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit	Can variable be changed?
Mass flow (Default: 0)	$\dot{m}$	kg/s	Yes
Specific enthalpy (Default: 100)	h	kJ/kg	Yes

The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Pressure	p	bar
Density	r	kg/m³
Temperature	T	°C
Specific enthalpy	h	kJ/kg

NetParam – network parameter assignment

Symbol



Function

The *NetParam* component type is used for flow network parameter assignment. To do so, the component is added at any location in any branch of the flow network.

Parameters

Parameter name	Description	Unit	Default value
Medium	You can select "*Water/Steam*", "*Liquid*" or "*Ideal Gas*" as the medium in the flow network	–	*Water/Steam*
FactorMomentum	Momentum factor for the flow in the branches of the flow network	m	450.0
sCompressionGas	Specific compression module for the "*Water/Steam*" medium with density ρ < 500 kg/m³ or the "*Ideal Gas*" medium	bar/kg	10.0
sCompressionLiquid	Specific compression module for the "*Water/Steam*" medium with density ρ > 500 kg/m³ or the "*Liquid*" medium	bar/kg	100.0
FactorThermalGas	Enthalpy balancing factor for the "*Water/Steam*" medium with density ρ < 500 kg/m³ or the "*Ideal Gas*" medium	1/kg	100.0
FactorThermalLiquid	Enthalpy balancing factor for the "*Water/Steam*" medium with density ρ > 500 kg/m³ or the "*Liquid*" medium	1/kg	0.1
DensityLiquid	Media density; only valid for "*Liquid*" medium	kg/m³	997.337
sHeatCapGas	Specific heat capacity for gas; only valid for "*Ideal Gas*" medium	kJ/kgK	1.0
sHeatCapLiquid	Specific heat capacity for liquids; only valid for "*Liquid*" medium	kJ/kgK	4.18
GasConstant	Specific gas constant of the medium; only valid for "*Ideal Gas*" medium	kJ/kgK	0.287

Additional parameters

Parameter name	Description	Unit	Default value
PressureInit	Initialization value for the pressure in the internal nodes of the flow network	bar	1.0
sEnthalpyInit	Initialization value for the specific enthalpy in the internal nodes of the flow network	kJ/kg	100.0
SmoothTransition	When this additional parameter is set to True, the variables *sCompression* and *FactorThermal* are calculated with a density-dependent linear transfer function	–	False
TemperatureEnvironment	Ambient temperature	°C	20.0
FactorHeatExchangeEnv	Factor of proportionality for heat exchange of the medium in the internal nodes with the environment; there is no heat exchange when the value is zero	kW/K	0.0

BranchParam – branch parameter assignment

Symbol



Function

The *BranchParam* component type is used for branch parameter assignment in the flow network. The components are placed anywhere along the branch to be assigned parameters.

Parameters

Parameter name	Description	Unit	Default value
FactorMomentum	Momentum factor for the flow in the branches of the flow network	m	450.0

Joint – joint

Symbol



Function

With the component type *Joint*, three branches connected to its connectors *A*, *B* and *C* can be combined in a node. An internal node is added to the flow network with the component *Joint* .

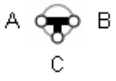
Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure	p	bar
Specific enthalpy	h	kJ/kg
Density	r	kg/m³
Temperature	T	°C

JointParam – parameterizable joint

Symbol



Function

With the component type *JointParam*, three branches connected to its connectors *A*, *B* and *C* can be combined in a node. An internal node is added to the flow network with the component *JointParam* .

Parameters

Parameter name	Description	Unit	Default value
sCompressionGas	Specific compression module for low-density media (gases/ vapors, ρ < 500 kg/m³)	bar/kg	10.0
sCompressionLiquid	Specific compression module for high density media (liquid, ρ > 500 kg/m³)	bar/kg	100.0
FactorThermalGas	Factor for enthalpy balance calculation of low-density media (gases/vapors, ρ < 500 kg/m³)	1/kg	100.0
FactorThermalLiquid	Factor for enthalpy balancing for high density media (liquid, ρ > 500 kg/m³)	1/kg	0.1

Additional parameters

Parameter name	Description	Unit	Default value
PressureInit	Initialization value for the pressure in the node	bar	1.0
sEnthalpyInit	Initialization value for the specific enthalpy in the node	kJ/kg	100.0
TemperatureEnvironment	Ambient temperature	°C	20.0
FactorHeatExchangeEnv	Proportionality factor for the heat exchange of the medium in the node with the environment; for a value of zero, no heat exchange occurs	kW/K	0.0

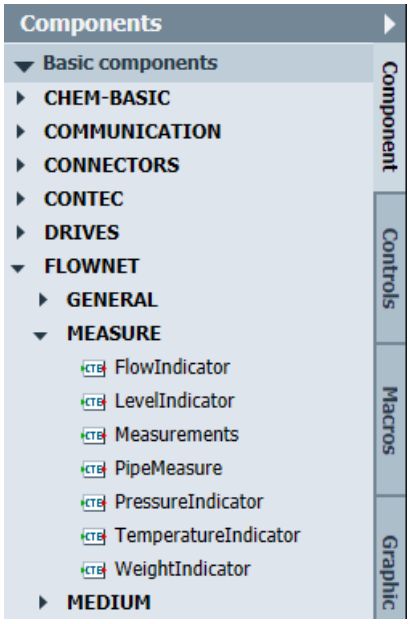
Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure	p	bar
Specific enthalpy	h	kJ/kg
Density	r	kg/m³
Temperature	T	°C

8.2.6.2 Measuring components

Components for simulating measurements in pipeline networks can be found in the *MEASURE* directory.



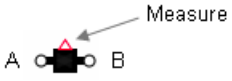
Connectors of the *Measure* connection type are used in the measuring components. The meaning of the individual signals for this type are given in the following table.

Table 8-40 Signals of the Measure connection type

Signal	Meaning	Unit
Temperature	Temperature of measurement	°C
Pressure	Pressure of measurement	bar
Level	Measured level	m
Weight	Measured weight	kg
Flow	Measured flow	kg/s

PipeMeasure – pipe measuring point

Symbol



Function

The *PipeMeasure* component type forms a measuring point in the tube. It is inserted with its connectors *A* and *B* at the desired measuring point in the flownet. The measuring process for the various variables is not simulated with suitable models; the variables calculated by the flownet solution procedure are output.

The measured variables are output over the *Measure* connector:

- absolute value $|\dot{m}|$ of the flow rate value,
- Pressure p_A at connector *A* and
- Temperature *T*.

All other signals of connector *Measure* are not set.

When the simulation is running, the direction of the medium flow is indicated by an arrow on the symbol.



Operating window

The following variables are displayed in the operating window:

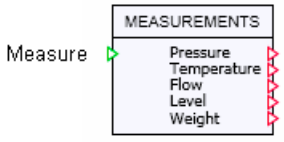
Variable	Symbol	Unit
Pressure	p	bar
Mass flow	$\dot{m}$	kg/s
Temperature	T	°C

The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Specific enthalpy	h	kJ/kg
Density	r	kg/m³

Measurements – measurement indicators

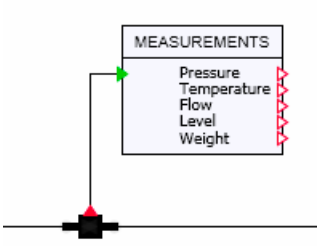
Symbol



Function

Component type *MeasureAll* returns the measured values bundled over measuring input *Measure* as individual signals at its outputs: pressure, temperature, flow, level and weight.

This type of component can be connected to the pipe measuring point, for example, and thereby output its measurement variables as individual signals.



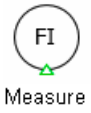
Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure	p	bar
Temperature	T	°C
Mass flow	$\dot{m}$	kg/s
Level	l	m
Weight	M	kg

FlowIndicator – flow measurement indicator

Symbol



Function

The *FlowIndicator* component type displays the flow specified using its *Measure* input.

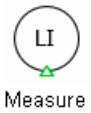
Operating window

The following variable is displayed in the operating window:

Variable	Symbol	Unit
Mass flow	$\dot{m}$	kg/s

LevelIndicator – level indicator

Symbol



Function

The *LevelIndicator* component type displays the level specified using its *Measure* input.

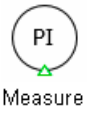
Operating window

The following variable is displayed in the operating window:

Variable	Symbol	Unit
Level	l	m

PressureIndicator – pressure indicator

Symbol



Function

The *PressureIndicator* component type displays the pressure specified using its *Measure* input.

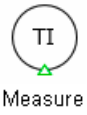
Operating window

The following variable is displayed in the operating window:

Variable	Symbol	Unit
Pressure	p	bar

TemperatureIndicator – temperature indicator

Symbol



Function

The *TemperatureIndicator* component type displays the temperature specified using its *Measure* input.

Operating window

The following variable is displayed in the operating window:

Variable	Symbol	Unit
Temperature	T	°C

WeightIndicator – weight indicator

Symbol



Function

The *WeightIndicator* component type displays the weight specified using its *Measure* input.

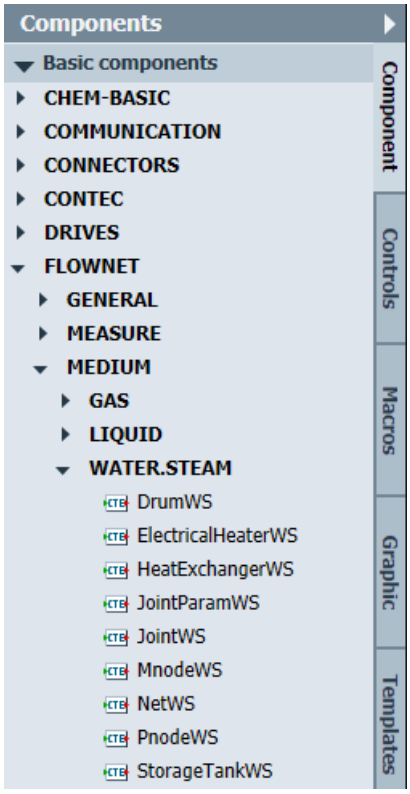
Operating window

The following variable is displayed in the operating window:

Variable	Symbol	Unit
Weight	M	kg

8.2.6.3 Component types for "water/steam" medium

Component types that can be used in the flownet with the water/steam medium are located in the *MEDIUM\WATER.STEAM* folder of the FLOWNET library. The medium parameter for a flownet that contains these components must be set to the value "*Water/Steam*".



NetWS – water/steam network parameter assignment

Symbol



Function

The *NetWS* component type is used for parameter assignment for a water/steam network. To do so, the component is added at any location in any branch of the flow network.

Parameters

Parameter name	Description	Unit	Default value
FactorMomentum	Momentum factor for the flow in the branches of the flow network	m	450.0
sCompressionSteam	Specific compression modulus for density ρ < 500 kg/m³ (steam)	bar/kg	10.0
sCompressionWater	Specific compression module for density ρ > 500 kg/m³ (water)	bar/kg	100.0
FactorThermalSteam	Factor for enthalpy balancing for density ρ < 500 kg/m³ (steam)	1/kg	100.0
FactorThermalWater	Factor for enthalpy balancing for density ρ > 500 kg/m³ (water)	1/kg	0.1

Additional parameters

Parameter name	Description	Unit	Default value
PressureInit	Initialization value for the pressure in the internal nodes of the flow network	bar	1.0
sEnthalpyInit	Initialization value for the specific enthalpy in the internal nodes of the flow network	kJ/kg	100.0
SmoothTransition	When this additional parameter is set to True, the variables *sCompression* and *FactorThermal* are calculated with a density-dependent linear transfer function	–	False
TemperatureEnvironment	Ambient temperature	°C	20.0
FactorHeatExchangeEnv	Factor of proportionality for heat exchange of the medium in the internal nodes with the environment; there is no heat exchange when the value is zero	kW/K	0.0

PnodeWS – water/steam pressure setting

Symbol



Function

The *PnodeWS* component type specifies the values for pressure *p* and specific enthalpy *h* at its connector A. This type of component forms a boundary for the flownet. When the flownet is represented as a graph, it corresponds to an (external) node for which the pressure and specific enthalpy are predefined.

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit	Can variable be changed?
Pressure (Default: 1)	p	bar	Yes
Specific enthalpy (Default: 100)	h	kJ/kg	Yes

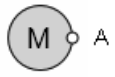
The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Mass flow	$\dot{m}$	kg/s
Density	r	kg/m³
Temperature	T	°C
Specific enthalpy	h	kJ/kg

For outflow $\dot{m} > 0$, for inflow $\dot{m} < 0$.

MnodeWS – water/steam mass flow setting

Symbol



Function

The *MnodeWS* component type specifies the values for mass flow $\dot{m}$ and specific enthalpy *h* at its connector A. This type of component forms a boundary for the flow network. When looking at the flow network as a graph, this corresponds to an inflow or outflow into an (internal) node or branch. An internal node with defined inflow or outflow is added to the flow network for each component of this type.

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit	Can variable be changed?
Mass flow (Default: 0)	$\dot{m}$	kg/s	Yes
Specific enthalpy (Default: 100)	h	kJ/kg	Yes

The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Pressure	p	bar
Density	r	kg/m³
Temperature	T	°C
Specific enthalpy	h	kJ/kg

JointWS – water/steam joint

Symbol



Function

With the component type *JointWS*, three branches connected to its connectors *A*, *B* and *C* can be combined in a node. An internal node is added to the flow network with the component *JointWS*.

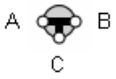
Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure	p	bar
Specific enthalpy	h	kJ/kg
Density	r	kg/m³
Temperature	T	°C

JointParamWS – water/steam parameterizable joint

Symbol



Function

With the component type *JointParamWS*, three branches connected to its connectors *A*, *B* and *C* can be combined in a node. An internal node is added to the flow network with the component *JointParamWS*.

Parameters

Parameter name	Description	Unit	Default value
sCompressionSteam	Specific compression modulus for density ρ < 500 kg/m³ (steam)	bar/kg	10.0
sCompressionWater	Specific compression module for density ρ > 500 kg/m³ (water)	bar/kg	100.0
FactorThermalSteam	Factor for enthalpy balancing for density ρ < 500 kg/m³ (steam)	1/kg	100.0
FactorThermalWater	Factor for enthalpy balancing for density ρ > 500 kg/m³ (water)	1/kg	0.1

Additional parameters

Parameter name	Description	Unit	Default value
PressureInit	Initialization value for the pressure in the node	bar	1.0
sEnthalpyInit	Initialization value for the specific enthalpy in the node	kJ/kg	100.0
TemperatureEnvironment	Ambient temperature	°C	20.0
FactorHeatExchangeEnv	Proportionality factor for the heat exchange of the medium in the node with the environment; for a value of zero, no heat exchange occurs	kW/K	0.0

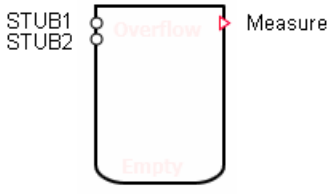
Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure	p	bar
Specific enthalpy	h	kJ/kg
Density	r	kg/m³
Temperature	T	°C

StorageTankWS – water/steam storage tank

Symbol



Function

The *StorageTankWS* component type provides the simulation with an open water tank, which means a tank that is not sealed from the environment.

Inflows and outflows occur via the *STUBx* connectors on the tank. For each of the *N* connectors, a throttling effect defined by

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_V^2 \, \rho \frac{\text{kg}}{\text{m}^3}} 12960 \left(\frac{\text{sec}}{\text{h}} \right)^2$$

is assumed. The tank can have a minimum of one and a maximum of 16 connectors. The connectors can be moved to any position on the outline of the component symbol by using the mouse while simultaneously pressing ALT.

It is assumed that the water in the tank is immediately completely mixed, which means a homogenous medium with an overall consistent density and enthalpy in the tank.

The inflows and outflows of water are balanced across the *N* connectors of the tank. The balanced mass *M* of the *N* flow rates $\dot{m}_i$ of the medium in the tank is thus defined by

$$\frac{dM}{dt} = \sum_{i=1}^{N} \dot{m}_i$$

The water volume balancing via the volume flows results in the rate of change of density ρ given by

$$\frac{d\rho}{dt} = \frac{\rho}{M} \left(\sum_{i=1, i \in Z}^{N} \dot{m}_i - \rho \sum_{i=1, i \in Z}^{N} \frac{1}{\rho_i} \dot{m}_i \right)$$

where only the inflows ($i \in Z$) need to be summed.

The specific enthalpy of the water is determined from the enthalpy balance according to

$$\frac{dh}{dt} = \frac{1}{M} \left(\sum_{i=1, i \in Z}^{N} h_i \dot{m}_i - h \sum_{i=1, i \in Z}^{N} \frac{1}{\rho_i} \dot{m}_i \right)$$

. Here, too, only the inflows ($i \in Z$) need to be summed.

The dynamic behavior of the water in the tank is described by these balances for mass *M*, density ρ and specific enthalpy *h*.

The variables calculated for the level *l* of water in the drum, its mass *M* and temperature *T* and the pressure p_0 at the floor of the drum are output at the *Measure* connector. The temperature *T* is calculated using the equation of state

$T = T(\rho, h)$

from the density and specific enthalpy of the water. The pressure *p* on the tank base is calculated as the sum of the weight pressure $\rho g l$ of the water and the atmospheric pressure p_U according to

$p = p_U + \rho g l$.

Borderline case "empty tank"

The outflow for an empty drum is severely throttled. The flow coefficient is thereby set to the value k_{v0} for maximum throttling of all connectors through which water flows out.

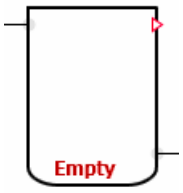
The tank is considered empty when the tank fill level V is lower than a specified minimum tank filling V_{min}:

$V < V_{min}$

When the tank is empty, balancing of the states is stopped, which means changes to the three state variables mass, density and specific enthalpy are rejected. The empty state continues until there is a sufficient increase in the fill level. In each increment the validity of

$$M \geq V_{min}\,\rho\left(1+\frac{H_{min}}{100\%}\right)$$

is checked. The required increase can be set with the hysteresis H_{min}. A corresponding indicator for an empty tank is shown in the component symbol.

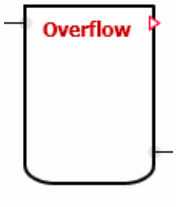


Borderline case "full tank"

For a full tank, the balanced contents are limited in the calculation of the state variables according to:

$M = V\rho$

A corresponding indicator is shown in the symbol.



Note

It is not checked whether the state variables (pressure, enthalpy) for the inflows and tank contents always have values for the water state.

In the simulation, inflows specified accordingly may result in a situation where the state values for the tank contents describe a water-steam mixture or pure steam, which has no physical meaning.

Parameters

Parameter name	Description	Unit	Default value
Volume	Volume *V* of the tank; can be adjusted online	m^3	10.0
Height	Height of the tank; can be adjusted online	m	5.0
NbrOfStubs	Number *N* of connectors	–	1

Additional parameters

Parameter name	Description	Unit	Default value
PressureOutside	Atmospheric pressure p_U; can be adjusted online	bar	1.0
LevelInit	Initialization value for the level	%	50.0
EnthalpyInit	Initialization value for the specific enthalpy in the contents	kJ/kg	100.0
DensityInit	Initialization value for the density of the contents	kg/m^3	997.337
Kvs	Uniform flow coefficient k_V for all connectors	m^3/h	360.0
Kv0	Flow coefficient k_{V0} for severe throttling in the tank connector	m^3/h	0.000001
MinVolume	Minimum tank volume V_{min}; can be adjusted online	m^3	0.001
MinVolumeHys	Hysteresis H_{min}; can be adjusted online	%	50.0

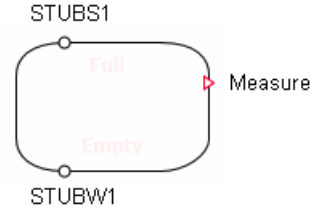
Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Level	l	m
Pressure	p	bar
Temperature	T	°C
Weight	M	kg

DrumWS – water/steam drum

Symbol



Function

The *DrumWS* component type provides the simulation with a drum, which means a closed container for water/steam that is sealed off from the environment.

The drum is assumed to be a cylindrical, horizontal or vertically standing container, whose inflows and outflows occur via the *STUBWx* and/or *STUBSx* connectors. For each of the connectors, a throttling effect defined by

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_V^2 \, \rho \frac{\text{kg}}{\text{m}^3}} 12960 \left(\frac{\text{sec}}{\text{h}} \right)^2$$

is assumed. At the N_W connectors *STUBWx*, medium with the state variables of the saturated water can be drawn out and medium with the state variables of saturated steam can be drawn out at the N_S connector *STUBSx*. The drum can have a minimum of one and a maximum of eight connectors of each type. The connectors can be moved to any position on the outline of the component symbol by using the mouse while simultaneously pressing ALT.

It is assumed that the medium in the drum immediately separates into two homogenous saturated phases: the saturated liquid phase and the saturated vapor phase.

The inflows and outflows of water/steam are balanced via the $N = N_W + N_S$ drum connectors. This results in a mass of water and steam in the drum, balanced via the mass flows, given by

$$\frac{dM}{dt} = \sum_{i=1}^{N} \dot{m}_i$$

or given

$M = \rho V$

the change of the average density to

$$V \frac{d\rho}{dt} = \sum_{i=1}^{N} \dot{m}_i$$

The energy is further balanced according to

$$\frac{d(hM)}{dt} = \sum_{i=1}^{N} h_i \dot{m}_i$$

Whereby h is the specific enthalpy (mean enthalpy) of water/steam in the drum and h_i is the specific enthalpy of the inflow and outflow at the i-th connector. The specific enthalpy of the medium in the drum must be applied for outflows, which means h'' for outflows of saturated water and h'' for outflows of saturated steam. The balancing is calculated as follows

$$M\frac{dh}{dt} = \sum_{i=1, i\in Z}^{N}(h_i - h)\dot{m}_i + (h' - h)\sum_{i=1, i\in A'}^{N}\dot{m}_i + (h'' - h)\sum_{i=1, i\in A''}^{N}\dot{m}_i$$

Whereby the inflows Z, outflows of saturated water A' and/or outflows of saturated steam A'' are summed.

In order to model a heat exchange with an ideally insulated drum, the enthalpy balance equation is extended as follows

$$M\frac{dh}{dt} = \sum_{i=1, i\in Z}^{N}(h_i - h)\dot{m}_i + (h' - h)\sum_{i=1, i\in A'}^{N}\dot{m}_i + (h'' - h)\sum_{i=1, i\in A''}^{N}\dot{m}_i + A\alpha(T_T - T_S)$$

T_T is the temperature of the drum wall, T_S is the saturation temperature of water/steam.

The heat storage in the drum wall is described by the heat balance

$$\frac{dT_T}{dt} = \frac{1}{M_T c_T} A\alpha(T_S - T_T)$$

.

The water level l, the saturation temperature T_S, the saturation pressure p_S and the mass M of water/steam are output at the *Measure* connector.

The STUBWx connectors are assumed to be located at the bottom of the drum. The pressure at these connectors therefore is the sum

$p = p'gl + p_S$

of the saturated steam and the hydrostatic pressure of the water.

Borderline case "empty drum"

When the drum is empty, the outflow is strongly throttled. The flow coefficient is thereby set to the value k_{V0} for maximum throttling of all connectors through which water or steam flows out.

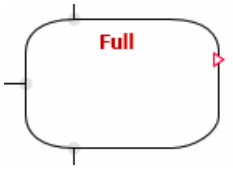
The drum is considered empty when the fill level $M_{W/\rho}$ is lower than a specified minimum fill level V_{min}:

$M_W < V_{min}\rho$.

When the drum is empty, balancing of the states is stopped, which means changes to the density and specific enthalpy are rejected. The empty state continues until there is a sufficient increase in the amount of water. In each increment the validity of

$$M_W \geq V_{min}\,\rho\left(1+\frac{H_{min}}{100\%}\right)$$

is checked. The required increase can be set with the hysteresis H_{min}. A corresponding indicator for an empty drum is shown in the component symbol.



Borderline case "full drum"

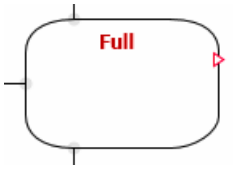
The drum is regarded as full when the amount of water reaches the maximum possible value:

$M_W < (V - V_{min})\,\rho$.

When the drum is full, balancing of the states is stopped, which means changes to the density and specific enthalpy are rejected. The full state continues until there is a sufficient decrease in the amount of water. In each increment the validity of

$$M_W \leq \rho V - \rho V_{min}\left(1+\frac{H_{min}}{100\%}\right)$$

is checked. The required decrease can be set with the hysteresis H_{min}. A corresponding indicator for a full drum is shown in the component symbol.



Parameters

Parameter name	Description	Unit	Default value
NbrOfStubsW	Number N_W the *STUBWx* connectors	–	1
NbrOfStubsS	Number N_S of *STUBSx* connectors	–	1
Position	Position of the drum: *Vertically* or *Horizontally*	–	*Vertically*
VolumeDrum	Volume *V* of the drum; can be adjusted online	m^3	10.0
HeightOrLength	Height or length of the drum; can be adjusted online	m	5.0
MassDrum	Mass M_T of the drum; can be adjusted online	kg	5000.0
SurfaceDrum	Inner surface *A* of the drum; can be adjusted online	m^2	12.5

Parameter name	Description	Unit	Default value
sHeatCapDrum	Specific heat capacity c_T of the drum; can be adjusted online	kJ/kgK	0.5
HeatTransCoe	Heat transfer coefficient *α* for water/ steam of the drum; can be adjusted online	kW/m²K	4.0

Additional parameters

Parameter name	Description	Unit	Default value
VolumeInit	Initialization value for the volume of water	%	50.0
TemperatureInit	Initialization value for the temperature of water/steam (saturation temperature)	°C	20.0
Kvs	Uniform flow coefficient k_V for all connectors	m³/h	360.0
Kv0	Flow coefficient k_{V0} for severe throttling in the drum connector	m³/h	0.00001
MinVolume	*Minimal drum v*olume V_{min}; can be adjusted online	m³	0.01
MinVolumeHys	Hysteresis H_{min}; can be adjusted online	%	50.0

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit	Can variable be changed?
Water level	L	m	Yes[1]
Saturation temperature	T	°C	Yes[1]
Weight of water/steam	M	kg	Yes[1]
Saturation pressure	p_B	bar	No
Water pressure	p_G	bar	No
Container temperature	T_{Drum}	°C	No

[1] Enable input with "Set"

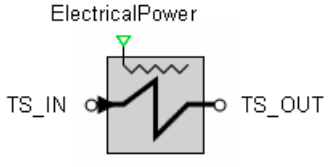
The following variables are displayed in the extended operating window:

- Mix: Intermediate variables
- Water: Variables for saturated water
- Steam: Variables for saturated steam

Variable	Symbol	Unit
Density	r	kg/m³
Specific enthalpy	h	kJ/kg

ElectricalHeaterWS – electrically heated heat exchanger for water/steam

Symbol



Function

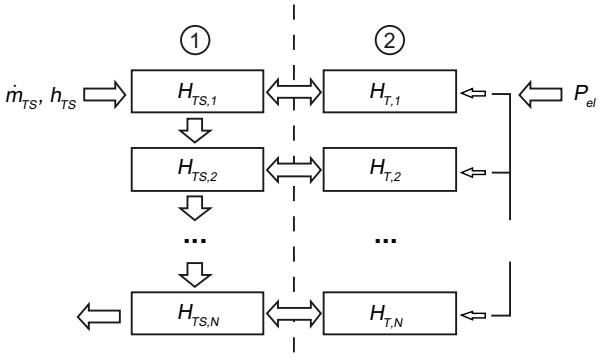
The *ElectricalHeaterWS* component type is used for simulation of an electric heat exchanger. The electrical heating power P_{el} is specified in kW with connector *ElectricalPower*.

Water/steam is directed via the *TS_IN* and *TS_OUT* connectors as a heated medium. The flow $\dot{m}$ is throttled according to the relationship

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_V^2 \, \rho_{TS} \frac{\text{kg}}{\text{m}^3}} 12900 \left(\frac{\text{sec}}{\text{h}} \right)^2$$

with throttle coefficient k_V. The reference direction for the flow is chosen as from *TS_IN* to *TS_OUT*.

For the heat transfer, a simple one tube model divided into segments is applied. The number *N* of segments can be set to a value between 4 and 16.



① Tube side
② Tube

It is assumed that the supplied electrical energy is completely converted into heat. The heat balance for a segment *i* of the spare tube with extremely simplified heat transfer is then given by

$$\frac{dT_{T,i}}{dt} = a_T P_{el} + b_T \left(T_{TS,i} - T_{T,i} \right)$$

with

$$a_T = \frac{1}{M_T c_T}, \quad b_T = \frac{A_{TS}\alpha_{TS}}{M_T c_T}$$

In addition, the water/steam-side enthalpy balancing for a segment *i* is described by

$$\frac{dh_{TS,i}}{dt} = a_{TS}\left(h_{TS,i-1} - h_{TS,i}\right) + b_{TS}\left(T_{T,i} - T_{TS,i}\right)$$

with

$$a_{TS} = \frac{N\dot{m}}{\rho_{TS} V_{TS}}, \quad b_{TS} = \frac{A_{TS}\alpha_{TS}}{\rho_{TS} V_{TS}}$$

$T_{T,i}$ and $T_{TS,i}$ are the temperatures of the tube segments and of the medium in the segments, and $h_{T,i}$ and $h_{TS,i}$are the corresponding specific enthalpies.

Parameters

Parameter name	Description	Unit	Default value
NbrOfSegments	Number *N* of segments: 4 ≤ *N* ≤ 16	–	4
Kvs	Throttle coefficient k_V	m^3/h	360.0
Volume	Volume V_{TS} of the medium (inner tube volume)	m^3	2.25
Surface	Surface A_{TS} of the tube on the water/ steam side	m^2	550.0
HeatTransCoef	Heat transfer coefficient α_{TS} of the tube to water/steam	kW/m^2K	4.0
sHeatCapTube	Specific heat capacity c_T of the tube	kJ/kgK	1.3
MassTube	Mass M_T of the tube	kg	10000.0

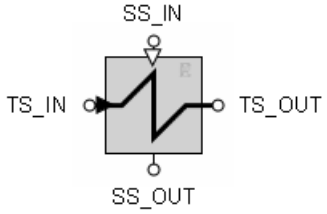
Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Flow	$\dot{m}$	kg/s
Pressure drop	Δp	bar
Enthalpy difference	Δh	kJ/kg
Temperatures for first and last segment, separated by tube and the heated medium	T_1 / T_N	°C
Electrical heat output	P_{el}	kW

HeatExchangerWS – heat exchanger water/steam to water/steam

Symbol



Function

The *HeatExchangerWS* component type is used for simulation of a heat exchanger for water/steam at the tube and shell side. The simulation is implemented for the three types:

- Parallel-flow heat exchanger
- Counter-flow heat exchanger
- Cross-flow heat exchanger

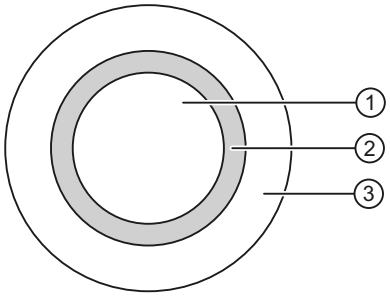
Both media are routed on the tube side via the connectors *TS_IN* and *TS_OUT*, and on the shell side via the connectors *SS_IN* and *SS_OUT*.

The flow $\dot{m}$ is throttled on the tube side and shell side according to the relationship

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_V^2 \, \rho \frac{\text{kg}}{\text{m}^3}} 12900 \left(\frac{\text{sec}}{\text{h}} \right)^2$$

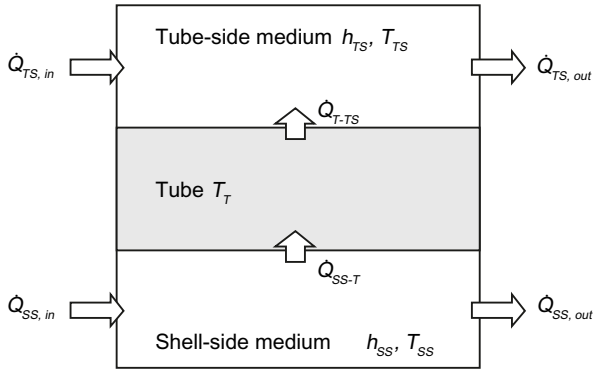
with the relevant throttle coefficient k_v. The chosen reference direction for the flow is from connector *_IN* to *_OUT* on the tube side and shell side.

For the heat transfer, a simple one tube model divided into segments is applied. The number *N* of segments can be set to a value between 4 and 16.



① Tube-side space
② Tube
③ Shell-side space

The water/steam heat on the tube and shell side and the heat in the tube itself are balanced.



The two heat transfers, from the shell side medium to the tube and from the tube to the tube side medium, are set according to

$$\dot{Q}_{SS-T} = A_{SS}\dot{q}_{SS-T} = A_{SS}\alpha_{SS}\left(T_{SS} - T_T\right)$$

$$\dot{Q}_{T-TS} = A_{TS}\dot{q}_{T-TS} = A_{TS}\alpha_{TS}\left(T_T - T_{TS}\right)$$

in an extremely simplified manner. For each segment *i* the heat balances

$$\frac{dh_{TS,i}}{dt} = a_{TS}\left(h_{TS,i-1} - h_{TS,i}\right) + b_{TS}\left(T_{T,i} - T_{TS,i}\right)$$

$$\frac{dT_{T,i}}{dt} = a_T\left(T_{SS,i} - T_{T,i}\right) + b_T\left(T_{TS,i} - T_{T,i}\right)$$

$$\frac{dh_{SS,i}}{dt} = a_{SS}\left(h_{SS,i-1} - h_{SS,i}\right) + b_{SS}\left(T_{T,i} - T_{SS,i}\right)$$

apply with the following coefficients, which are the same for all segments:

$$a_{TS} = \frac{N\dot{m}_{TS}}{\rho_{TS}V_{TS}}, \quad b_{TS} = \frac{A_{TS}\alpha_{TS}}{\rho_{TS}V_{TS}}, \quad a_{SS} = \frac{N\dot{m}_{SS}}{\rho_{SS}V_{SS}}, \quad b_{SS} = \frac{A_{SS}\alpha_{SS}}{\rho_{SS}V_{SS}}, \quad a_T = \frac{A_{SS}\alpha_{SS}}{M_T c_T}, \quad b_T = \frac{A_{TS}\alpha_{TS}}{M_T c_T}$$

The values set for the media in the shell side and tube side of the flownet apply to the densities ρ_{TS}, ρ_{SS} and the specific heat capacities c_{TS}, c_{SS}.

For initialization, the temperatures of the tube segments are set to the temperature of the tube-side medium, calculated on the basis of the pressure (input *FNTS.PRESSURE*) and the specific enthalpy (input *FNTS.HSPEC*).

Parameters

Parameter name	Description	Unit	Default value
Type	Type of heat exchanger: *ParallelFlow*, *CounterFlow*, *CrossFlow*	–	*ParallelFlow*
NbrOfSegments	Number *N* of segments: $4 \le N \le 16$	–	4
KvsSS	Shell side throttle coefficient k_V	m^3/h	360.0
KvsTS	Tube side throttle coefficient k_V	m^3/h	360.0
VolumeSS	Shell side volume V_{SS}	m^3	1.65
VolumeTS	Tube side volume V_{TS}	m^3	2.25
SurfaceSS	Shell-side surface A_{SS} of the tube (exterior surface of the tube)	m^2	720.0
SurfaceTS	Tube side surface A_{TS} of the tube (interior surface of the tube)	m^2	550.0
HeatTransCoefSS	Heat transfer coefficient α_{SS} on the tube exterior	kW/m^2K	4.0
HeatTransCoefTS	Heat transfer coefficient α_{TS} on the tube interior	kW/m^2K	4.0
sHeatCapTube	Specific heat capacity c_T of the tube	kJ/kgK	1.3
MassTube	Mass M_T of the tube	kg	10000.0

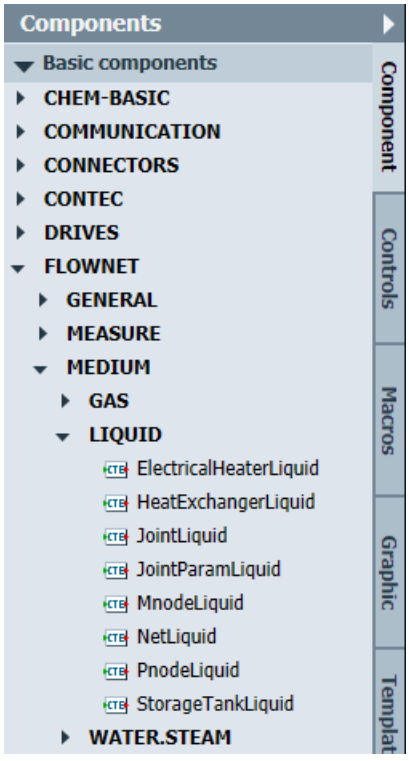
Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Flow	$\dot{m}$	kg/s
Pressure drop	Δp	bar
Enthalpy difference	Δh	kJ/kg
Temperatures for first and last segment, by • Tube • Medium (shell side) • Medium (tube side)	T_1 / T_N	°C

8.2.6.4 Component types for liquid medium

Component types that can be used in the flownet with the liquid medium are located in the directory *MEDIUM\LIQUID* of the FLOWNET library. The medium parameter for a flownet that contains these components should be set to the value "*Liquid*", if applicable.



NetLiquid – liquid network parameter assignment

Symbol



Function

The *NetLiquid* component type is used for parameter assignment for a liquid network. To do so, the component is added at any location in any branch of the flow network.

Parameters

Parameter name	Description	Unit	Default value
Density	Density of the liquid	kg/m³	997.337
sHeatCapacity	Specific heat capacity of the liquid	kJ/kgK	4.18
FactorMomentum	Momentum factor for the flow in the branches of the flow network	m	450.0
sCompression	Specific compression modulus of the liquid	bar/kg	100.0
FactorThermal	Factor for enthalpy balancing	1/kg	0.1

Additional parameters

Parameter name	Description	Unit	Default value
PressureInit	Initialization value for the pressure in the internal nodes of the flow network	bar	1.0
TemperatureInit	Initialization value for the temperature in the internal nodes of the flownet	°C	20.0
TemperatureEnvironment	Ambient temperature	°C	20.0
FactorHeatExchangeEnv	Factor of proportionality for heat exchange of the medium in the internal nodes with the environment; there is no heat exchange when the value is zero	kW/K	0.0

PnodeLiquid – liquid pressure setting

Symbol



Function

The *PnodeLiquid* component type specifies the values for pressure and temperature at its connector. It forms a boundary for the flownet to which it is connected. When you look at the flow network as a graph, *PnodeLiquid* represents an (external) node for which the pressure and temperature specifications are fixed.

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit	Can variable be changed?
Pressure (Default: 1)	p	bar	Yes
Temperature (Default: 20)	T	°C	Yes

The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Mass flow	$\dot{m}$	kg/s
Density	r	kg/m³
Temperature	T	°C

For outflow $\dot{m} > 0$, for inflow $\dot{m} < 0$.

MnodeLiquid – liquid mass flow setting

Symbol



Function

The *MnodeLiquid* component type specifies the values for mass flow and temperature at its connector. It forms a boundary for the flownet to which it is connected. When you look at the flow network as a graph, *MnodeLiquid* represents an inflow or outflow into an (internal) node or branch. Thus an internal node with a defined inflow or outflow is added to the flownet.

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit	Can variable be changed?
Mass flow (Default: 0)	$\dot{m}$	kg/s	Yes
Temperature (Default: 20)	T	°C	Yes

The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Pressure	p	bar
Density	r	kg/m³
Temperature	T	°C

JointLiquid – liquid joint

Symbol



Function

With the component type *JointLiquid*, three branches connected to its connectors can be combined in a node. All connectors are equal. An internal node is added to the flow network with the component *JointLiquid*.

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure	p	bar
Temperature	T	°C

JointParamLiquid – liquid parameterizable joint

Symbol



Function

With the component type *JointParamLiquid*, three branches connected to its connectors can be combined in a node. All connectors are equal. An internal node is added to the flow network with the component *JointParamLiquid*. This node can be assigned parameters.

Parameters

Parameter name	Description	Unit	Default value
sHeatCapacity	Specific heat capacity of the liquid	kJ/kgK	4.18
sCompression	Specific compression modulus of the liquid	bar/kg	100.0
FactorThermal	Factor for enthalpy balancing	1/kg	0.1

Additional parameters

Parameter name	Description	Unit	Default value
PressureInit	Initialization value for the pressure in the node	bar	1.0
TemperatureInit	Initialization value for the temperature in the node	°C	20.0
TemperatureEnvironment	Ambient temperature	°C	20.0
FactorHeatExchangeEnv	Factor of proportionality for heat exchange of the medium in the node with the environment; there is no heat exchange when the value is zero	kW/K	0.0

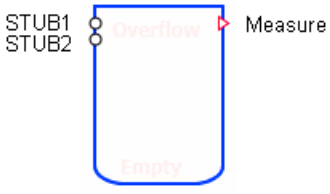
Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure	p	bar
Temperature	T	°C

StorageTankLiquid – liquid storage tank

Symbol



Function

The *StorageTankLiquid* component type provides the simulation with an open tank for liquids, which means a tank that is not sealed from the environment.

Inflows and outflows occur via the *STUBx* connectors on the tank. For each of the *N* connectors, a throttling effect defined by

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_V^2 \, \rho \frac{\text{kg}}{\text{m}^3}} 12900 \left(\frac{\text{sec}}{\text{h}} \right)^2$$

is assumed. The tank can have a minimum of one and a maximum of 16 connectors. The connectors can be moved to any position on the outline of the component symbol by using the mouse while simultaneously pressing ALT.

It is assumed that the liquid in the tank is immediately completely mixed, which means it is a homogenous medium with an overall consistent density and temperature.

The inflows and outflows are balanced across the *N* connectors of the tank. The balanced mass *M* of the *N* mass flows $\dot{m}_i$ of the liquid in the tank, is defined by

$$\frac{dM}{dt} = \sum_{i=1}^{N} \dot{m}_i$$

The temperature of the liquid is balanced as a mixing temperature from the inflows ($i \in Z$) according to

$$\frac{dT}{dt} = \frac{1}{M} \left(\frac{1}{c_L} \sum_{i=1, i \in Z}^{N} h_i \dot{m}_i - T \sum_{i=1, i \in Z}^{N} \frac{1}{\rho_i} \dot{m}_i \right)$$

.

The dynamic behavior of the liquid in the tank is described by these balances for mass *M* and temperature *T*.

The calculated variables for the level *l* of the liquid in the tank, its mass *M* and temperature *T* as well as pressure p_0 on the tank base are output at the *Measure* connector. The pressure *p* on the tank base is calculated from the weight pressure $\rho g l$ of the liquid and the atmospheric pressure p_U according to

$p = p_U + \rho g l$

Borderline case "empty tank"

The outflow for an empty drum is severely throttled. The flow coefficient is thereby set to the value k_{v0} for maximum throttling of all connectors through which the medium flows out.

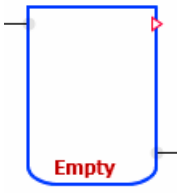
The tank is considered empty when the tank fill level *V* is lower than a specified minimum tank filling V_{min}:

$V < V_{min}$

When the tank is empty, balancing of the states is stopped, which means changes to the mass and temperature of the liquid are rejected. The empty state continues until there is a sufficient increase in the fill level. In each increment the validity of

$$M \geq V_{min}\,\rho\left(1+\frac{H_{min}}{100\%}\right)$$

is checked. The required increase can be set with the hysteresis H_{min}. A corresponding indicator for an empty tank is shown in the component symbol.

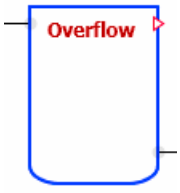


Borderline case "full tank"

For a full tank, only the balanced contents are limited in the calculation:

$M = V\rho$

A corresponding indicator is shown in the symbol.



Parameters

Parameter name	Description	Unit	Default value
Volume	Volume *V* of the tank; can be adjusted online	m^3	10.0
Height	Height of the tank; can be adjusted online	m	5.0
NbrOfStubs	Number *N* of connectors	–	1

Additional parameters

Parameter name	Description	Unit	Default value
PressureOutside	Atmospheric pressure p_U; can be adjusted online	bar	1.0
LevelInit	Initialization value for the level	%	50.0
TemperatureInit	Initialization value for the temperature of the contents	°C	20.0
Kvs	Uniform flow coefficient k_V for all connectors	m^3/h	360.0

Parameter name	Description	Unit	Default value
Kv0	Flow coefficient k_{V0} for severe throttling in the tank connector	m^3/h	0.000001
Density	Density ρ of the liquid in the tank	kg/m^3	997.337
sHeatCapacity	Specific heat capacity c_L of the liquid in the tank	kJ/kgK	4.18
MinVolume	Minimum tank volume V_{min}; can be adjusted online	m^3	0.001
MinVolumeHys	Hysteresis H_{min}; can be adjusted online	%	50.0

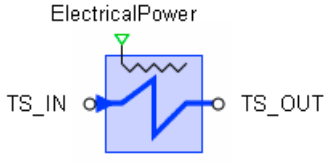
Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Level	l	m
Pressure	p	bar
Temperature	T	°C
Weight	M	kg

ElectricalHeaterLiquid – electrically heated heat exchanger for liquid

Symbol



Function

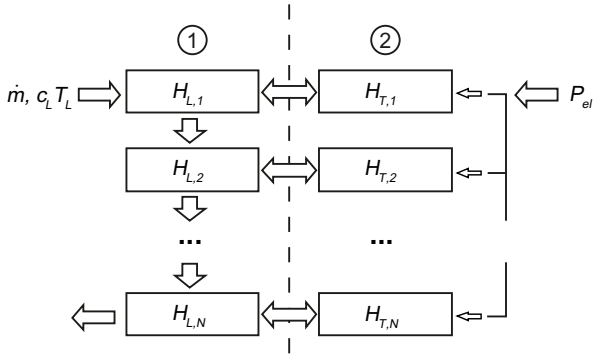
The *ElectricalHeaterLiquid* component type is used for simulation of an electric heat exchanger. The electrical heating power P_{el} is specified in kW with connector *ElectricalPower* .

The liquid is directed via the connectors *TS_IN* and *TS_OUT* as a heated medium. The flow $\dot{m}$ is throttled according to the relationship

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_V^2 \, \rho \frac{\text{kg}}{\text{m}^3}} 12900 \left(\frac{\text{sec}}{\text{h}} \right)^2$$

with throttle coefficient k_V. The reference direction for the flow is chosen as from *TS_IN* to *TS_OUT*.

For the heat transfer, a simple one tube model divided into segments is applied. The number *N* of segments can be set to a value between 4 and 16.



① Tube side
② Tube

It is assumed that the supplied electrical energy is completely converted into heat. The heat balance for a segment *i* of the spare tube with extremely simplified heat transfer is then given by

$$\frac{dT_{T,i}}{dt} = a_T P_{el} + b_T \left(T_{L,i} - T_{T,i}\right)$$

with

$$a_T = \frac{1}{M_T c_T}, \quad b_T = \frac{A\,\alpha}{M_T c_T}$$

In addition, the heat balance of the liquid in a segment *i* is given by

$$\frac{dT_{L,i}}{dt} = a\left(T_{L,i-1} - T_{L,i}\right) + b\left(T_{T,i} - T_{L,i}\right)$$

with

$$a = \frac{N\dot{m}}{\rho V}, \quad b = \frac{A\,\alpha}{c_L \rho V}$$

$T_{T,i}$ and $T_{L,i}$ are the temperatures of the tube segment and the liquid in the segment *i*. The values of the flow network apply for the density ρ and the specific heat capacity c_L of the liquid.

Parameters

Parameter name	Description	Unit	Default value
NbrOfSegments	Number *N* of segments: 4 ≤ *N* ≤ 16	–	4
Kvs	Throttle coefficient k_V	m^3/h	360.0
Volume	Volume *V* of the liquid (inner tube volume)	m^3	2.25
Surface	Surface *A* of the tube interior	m^2	550.0

Parameter name	Description	Unit	Default value
HeatTransCoef	Heat transfer coefficient α from tube to liquid	kW/m²K	4.0
sHeatCapTube	Specific heat capacity c_T of the tube	kJ/kgK	1.3
MassTube	Mass M_T of the tube	kg	10000.0

Additional parameters

Parameter name	Description	Unit	Default value
TemperatureInit	Temperature for liquid and tube when the heat exchanger is initialized	°C	20.0

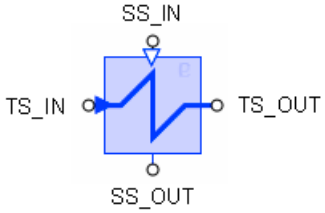
Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Flow	$\dot{m}$	kg/s
Pressure drop	Δp	bar
Temperature difference	ΔT	°C
Temperatures for first and last segment, separated by tube and the heated liquid	T_1/ T_N	°C
Electrical heat output	P_{el}	kW

HeatExchangerLiquid – heat exchanger liquid to liquid

Symbol



Function

The *HeatExchangerLiquid* component type is used for simulation of a heat exchanger for liquids at the tube and shell side. The simulation is implemented for the three types:

- Parallel-flow heat exchanger
- Counter-flow heat exchanger
- Cross-flow heat exchanger

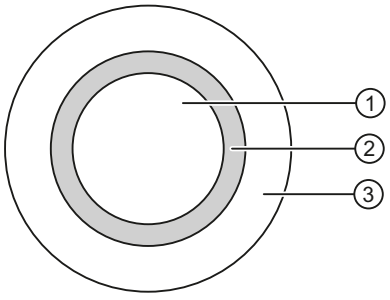
Both media are routed on the tube side via the connectors *TS_IN* and *TS_OUT*, and on the shell side via the connectors *SS_IN* and *SS_OUT*.

The flow $\dot{m}$ is throttled on the tube side and shell side according to the relationship

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_V^2 \, \rho_{TS} \frac{\text{kg}}{\text{m}^3}} 12900 \left(\frac{\text{sec}}{\text{h}} \right)^2$$

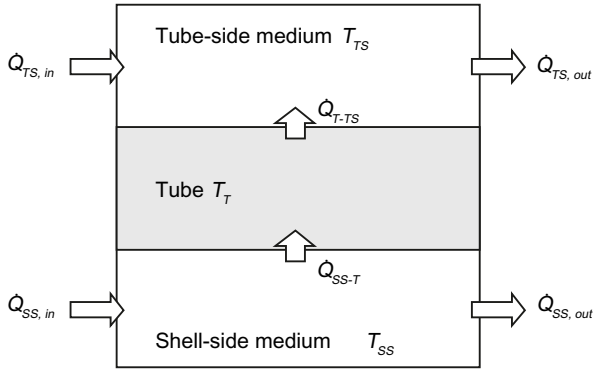
with the relevant throttle coefficient k_v. The chosen reference direction for the flow is from connector *_IN* to *_OUT* on the tube side and shell side.

For the heat transfer, a simple one tube model divided into segments is applied. The number *N* of segments can be set to a value between 4 and 16.



① Tube-side space
② Tube
③ Shell-side space

The liquid heat on the tube and shell side and the heat in the tube itself are balanced.



The two heat transfers, from the shell side liquid to the tube and from the tube to the tube side liquid, are set according to

$$\dot{Q}_{SS-T} = A_{SS}\dot{q}_{SS-T} = A_{SS}\alpha_{SS}\left(T_{SS} - T_T\right)$$

$$\dot{Q}_{T-TS} = A_{TS}\dot{q}_{T-TS} = A_{TS}\alpha_{TS}\left(T_T - T_{TS}\right)$$

in an extremely simplified manner. For each segment *i* the heat balances

$$\frac{dT_{TS,i}}{dt} = a_{TS}\left(T_{TS,i-1} - T_{TS,i}\right) + b_{TS}\left(T_{T,i} - T_{TS,i}\right)$$

$$\frac{dT_{T,i}}{dt} = a_{T}\left(T_{SS,i} - T_{T,i}\right) + b_{T}\left(T_{TS,i} - T_{T,i}\right)$$

$$\frac{dT_{SS,i}}{dt} = a_{SS}\left(T_{SS,i-1} - T_{SS,i}\right) + b_{SS}\left(T_{T,i} - T_{SS,i}\right)$$

apply with the following coefficients, which are the same for all segments:

$$a_{TS} = \frac{N\dot{m}_{TS}}{\rho_{TS}V_{TS}}, \quad b_{TS} = \frac{A_{TS}\alpha_{TS}}{\rho_{TS}V_{TS}}, \quad a_{SS} = \frac{N\dot{m}_{SS}}{\rho_{SS}V_{SS}}, \quad b_{SS} = \frac{A_{SS}\alpha_{SS}}{\rho_{SS}V_{SS}}, \quad a_{T} = \frac{A_{SS}\alpha_{SS}}{M_{T}c_{T}}, \quad b_{T} = \frac{A_{TS}\alpha_{TS}}{M_{T}c_{T}}$$

The values set for the liquid in the shell side and tube side of the flownet apply to the densities ρ_{TS}, ρ_{SS} and the specific heat capacities c_{TS}, c_{SS}.

For initialization, the temperatures of the tube segments are set to the temperature of the tube side medium, calculated from the specific enthalpy (*FNTS.HSPEC* input).

Parameters

Parameter name	Description	Unit	Default value
Type	Type of heat exchanger: *Parallel-Flow*, *CounterFlow*, *CrossFlow*	–	*ParallelFlow*
NbrOfSegments	Number *N* of segments: 4 ≤ *N* ≤ 16	–	4
KvsSS	Shell side throttle coefficient k_V	m^3/h	360.0
KvsTS	Tube side throttle coefficient k_V	m^3/h	360.0
VolumeSS	Shell side volume V_{SS}	m^3	1.65
VolumeTS	Tube side volume V_{TS}	m^3	2.25
SurfaceSS	Shell-side surface A_{SS} of the tube (exterior surface of the tube)	m^2	720.0
SurfaceTS	Tube side surface A_{TS} of the tube (interior surface of the tube)	m^2	550.0
HeatTransCoefSS	Heat transfer coefficient α_{SS} on the tube exterior	kW/m^2K	4.0
HeatTransCoefTS	Heat transfer coefficient α_{TS} on the tube interior	kW/m^2K	4.0
sHeatCapTube	Specific heat capacity c_T of the tube	kJ/kgK	1.3
MassTube	Mass M_T of the tube	kg	10000.0

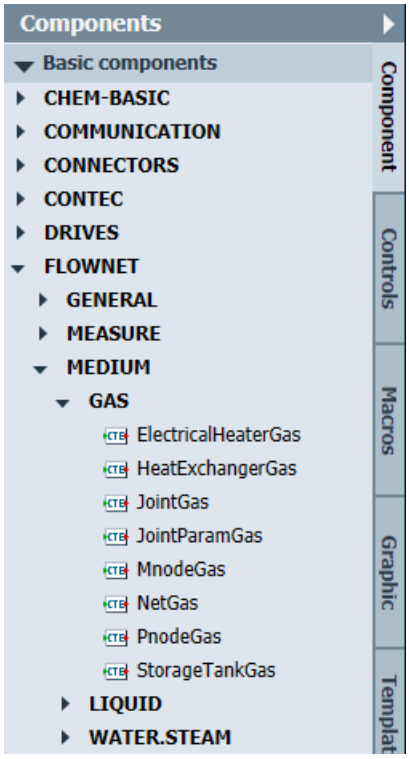
Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Flow	$\dot{m}$	kg/s
Pressure drop	Δp	bar
Temperature difference	ΔT	°C
Temperatures for first and last segment, by • Tube • Liquid (shell side) • Liquid (tube side)	T_1 / T_N	°C

8.2.6.5 Component types for gas medium

Component types that can be used in the flownet with the gas medium are located in the directory *MEDIUM\GAS* of the FLOWNET library. The medium parameter for a flownet that contains these components should be set to the value "*Ideal Gas*", if applicable.



NetGas – gas network parameter assignment

Symbol



Function

The *NetGas* component type is used for parameter assignment for a gas network. To do so, the component is added at any location in any branch of the flow network.

Parameters

Parameter name	Description	Unit	Default value
GasConstant	Specific gas constant	kJ/kgK	0.287
sHeatCapacity	Specific heat capacity of the gas	kJ/kgK	1.0
FactorMomentum	Momentum factor for the flow in the branches of the flow network	m	450.0
sCompression	Specific compression modulus of the gas	bar/kg	10.0
FactorThermal	Factor for enthalpy balancing	1/kg	100.0

Additional parameters

Parameter name	Description	Unit	Default value
PressureInit	Initialization value for the pressure in the internal nodes of the flow network	bar	1.0
TemperatureInit	Initialization value for the temperature in the internal nodes of the flownet	°C	20.0
TemperatureEnvironment	Ambient temperature	°C	20.0
FactorHeatExchangeEnv	Factor of proportionality for heat exchange of the medium in the internal nodes with the environment; there is no heat exchange when the value is zero	kW/K	0.0

PnodeGas – gas pressure setting

Symbol



Function

The *PnodeGas* component type specifies the values for pressure and temperature at its connector. It forms a boundary for the flownet to which it is connected. When you look at the flow network as a graph, *PnodeGas* represents an (external) node for which the pressure and temperature specifications are fixed.

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit	Can variable be changed?
Pressure (Default: 1)	p	bar	Yes
Temperature (Default: 20)	T	°C	Yes

The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Mass flow	$\dot{m}$	kg/s
Density	r	kg/m³
Temperature	T	°C

For outflow $\dot{m} > 0$, for inflow $\dot{m} < 0$.

MnodeGas – gas mass flow setting

Symbol



Function

The *MnodeGas* component type specifies the values for mass flow and temperature at its connector. It forms a boundary for the flownet to which it is connected. When you look at the flow network as a graph, *Mnode* represents an inflow or outflow into an (internal) node or branch. Thus an internal node with a defined inflow or outflow is added to the flownet.

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit	Can variable be changed?
Mass flow (Default: 0)	$\dot{m}$	kg/s	Yes
Temperature (Default: 20)	T	°C	Yes

The following variables are displayed in the extended operating window:

Variable	Symbol	Unit
Pressure	p	bar
Density	r	kg/m³
Temperature	T	°C

For outflow $\dot{m} > 0$, for inflow $\dot{m} < 0$.

JointGas – gas joint

Symbol



Function

With the component type *JointGas*, three branches connected to its connectors can be combined in a node. All connectors are equal. An internal node is added to the flow network with the component *JointGas*.

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure	p	bar
Temperature	T	°C
Density	r	kg/m³

JointParamGas – parameterizable joint

Symbol



Function

With the component type *JointParamGas*, three branches connected to its connectors can be combined in a node. All connectors are equal. An internal node is added to the flow network with the component *JointParamGas*. This node can be assigned parameters.

Parameters

Parameter name	Description	Unit	Default value
GasConstant	Specific gas constant	kJ/kgK	0.287
sHeatCapacity	Specific heat capacity of the gas	kJ/kgK	1.0
sCompression	Specific compression modulus of the gas	bar/kg	10.0
FactorThermal	Factor for enthalpy balancing	1/kg	100.0

Additional parameters

Parameter name	Description	Unit	Default value
PressureInit	Initialization value for the pressure in the node	bar	1.0
TemperatureInit	Initialization value for the temperature in the node	°C	20.0
TemperatureEnvironment	Ambient temperature	°C	20.0
FactorHeatExchangeEnv	Factor of proportionality for heat exchange of the medium in the node with the environment; there is no heat exchange when the value is zero	kW/K	0.0

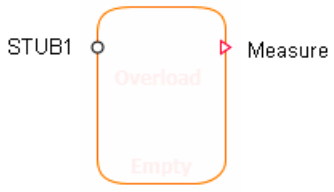
Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure	p	bar
Temperature	T	°C
Density	r	kg/m³

StorageTankGas – gas storage tank

Symbol



Function

The *StorageTankGas* component type provides the simulation of a gas tank.

Inflows and outflows occur via the *STUBx* connectors on the tank. For each of the *N* connectors, a throttling effect defined by

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_V^2 \, \rho \frac{\text{kg}}{\text{m}^3}} 12900 \left(\frac{\text{sec}}{\text{h}} \right)^2$$

is assumed. The tank can have a minimum of one and a maximum of 16 connectors. The connectors can be moved to any position on the outline of the component symbol by using the mouse while simultaneously pressing ALT.

It is assumed that the gas in the tank is immediately completely mixed, which means a homogenous medium in the tank with an overall consistent density and temperature in the tank.

The inflows and outflows are balanced across the *N* connectors of the tank. The balanced mass *M* of the *N* flow rates $\dot{m}_i$ of the gas in the tank, is defined by

$$\frac{dM}{dt} = \sum_{i=1}^{N} \dot{m}_i$$

The specific enthalpy *h* of the gas is balanced from the inflows ($i \in Z$) according to

$$\frac{dh}{dt} = \frac{1}{M} \left(\sum_{i=1, i \in Z}^{N} h_i \dot{m}_i - h \sum_{i=1, i \in Z}^{N} \dot{m}_i \right)$$

.

The dynamic behavior of the gas in the tank is described by these balances for mass *M* and specific enthalpy *h*.

At the connector *Measure*, the variables for pressure *p*, temperature

$$T = \frac{h}{c_G}$$

and mass M of the gas are output. The pressure is calculated from the equation for ideal gas:

$$p = \frac{R_s (T + 273{,}15\mathrm{K})\, M}{V \cdot 10^5 \frac{\mathrm{Pa}}{\mathrm{bar}}}$$

Borderline case "empty tank"

The outflow for an empty drum is severely throttled. The flow coefficient is thereby set to the value k_{v0} for maximum throttling of all connectors through which gas flows out.

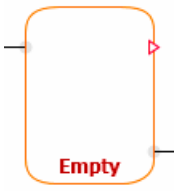
The tank is considered empty when the fill level M is lower than a specified minimum tank fill level M_{min}:

$M < M_{min}$

When the tank is empty, balancing of the states is stopped, which means changes to the mass and specific enthalpy of the gas are rejected. The empty state continues until there is a sufficient increase in the fill level. In each increment the validity of

$$M \geq M_{min} \left(1 + \frac{H_{min}}{100\%}\right)$$

is checked. The required increase can be set with the hysteresis H_{min}. A corresponding indicator for an empty tank is shown in the component symbol.



Borderline case "full tank"

In order to limit the pressure in the tank to meaningful values while the simulation is running, the tank is considered full when its pressure p reaches a specified maximum pressure p_{max}:

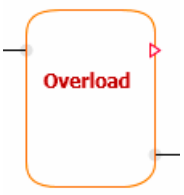
$p < p_{max}$

When the tank is full, the inflow is strongly throttled. The flow coefficient is thereby set to the value k_{v0} for maximum throttling of all connectors through which gas flows in.

When the tank is full, balancing of the states is stopped, which means changes to the mass and specific enthalpy of the gas are rejected. The full state continues until there is a sufficient decrease ΔM in the fill level. In each increment the validity of

$$\Delta M \geq M_{min} \left(1 + \frac{H_{min}}{100\%}\right)$$

is checked. The required decrease can be set with the hysteresis H_{min}. A corresponding indicator for a full tank is shown in the component symbol.



Parameters

Parameter name	Description	Unit	Default value
Volume	Volume *V* of the tank; can be adjusted online	m^3	10.0
NbrOfStubs	Number *N* of connectors	–	1

Additional parameters

Parameter name	Description	Unit	Default value
PressureInit	Initialization value for pressure; can be adjusted online	bar	1.0
TemperatureInit	Initialization value for the temperature of the gas	°C	20.0
PressureMax	Maximum tank pressure p_{max}; can be adjusted online	bar	100.0
Kvs	Uniform flow coefficient k_V for all connectors	m^3/h	36.0
Kv0	Flow coefficient k_{V0} for maximum throttling in the tank connector	m^3/h	0.000001
GasConstant	Specific gas constant R_S for the gas in the tank	kJ/kgK	0.287
sHeatCapacity	Specific heat capacity c_G of the gas in the tank	kJ/kgK	1.0
MinMass	Minimum fill quantity M_{min} of the tank; can be adjusted online	kg	0.015
MinMassHys	Hysteresis H_{min}; can be adjusted online	%	50.0

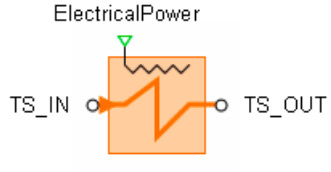
Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Pressure	p	bar
Temperature	T	°C
Mass of the gas	M	kg

ElectricalHeaterGas – electrically heated heat exchanger for gas

Symbol



Function

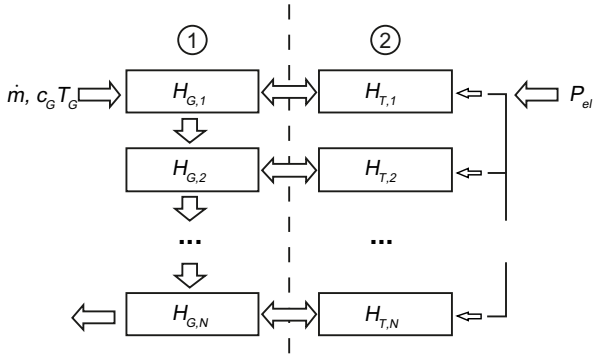
The *ElectricalHeaterGas* component type is used for simulation of an electric gas heat exchanger. The electrical heating power P_{el} is specified in kW with connector *ElectricalPower*.

The heated gas is directed via the connectors *TS_IN* and *TS_OUT*. The flow $\dot{m}$ is throttled according to the relationship

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_V^2 \, \rho_{TS} \frac{\text{kg}}{\text{m}^3}} 12900 \left(\frac{\text{sec}}{\text{h}}\right)^2$$

with throttle coefficient k_V. The reference direction for the flow is chosen as from *TS_IN* to *TS_OUT*.

For the heat transfer, a simple one tube model divided into segments is applied. The number *N* of segments can be set to a value between 4 and 16.



① Tube side
② Tube

It is assumed that the supplied electrical energy is completely converted into heat. The heat balance for a segment *i* of the spare tube with extremely simplified heat transfer is then given by

$$\frac{dT_{G,i}}{dt} = a_T P_{el} + b_T \left(T_{G,i} - T_{T,i}\right)$$

with

$$a_T = \frac{1}{M_T c_T}, \quad b_T = \frac{A\,\alpha}{M_T c_T}$$

In addition, the heat balance of the gas in a segment *i* is given by

$$\frac{dT_{G,i}}{dt} = a\left(T_{G,i-1} - T_{G,i}\right) + b\left(T_{T,i} - T_{G,i}\right)$$

with

$$a = \frac{N\,\dot{m}}{\rho V}, \quad b = \frac{A\,\alpha}{c_G \rho V}$$

$T_{T,i}$ and $T_{G,i}$ are the temperatures of the tube segment and the gas in segment *i*. The values of the flow network apply for the density ρ and the specific heat capacity c_G of the gas.

Parameters

Parameter name	Description	Unit	Default value
NbrOfSegments	Number *N* of segments: 4 ≤ *N* ≤ 16	–	4
Kvs	Throttle coefficient k_V	m^3/h	360.0
Volume	Volume *V* of the gas (inner tube volume)	m^3	2.25
Surface	Surface *A* of the tube interior	m^2	550.0
HeatTransCoef	Heat transfer coefficient *α* from tube to gas	kW/m^2K	0.1
sHeatCapTube	Specific heat capacity c_T of the tube	kJ/kgK	1.3
MassTube	Mass M_T of the tube	kg	1000.0

Additional parameters

Parameter name	Description	Unit	Default value
TemperatureInit	Temperature for liquid and tube when the heat exchanger is initialized	°C	20.0

Operating window

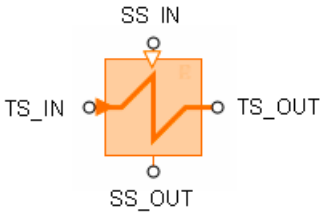
The following variables are displayed in the operating window:

Variable	Symbol	Unit
Flow	$\dot{m}$	kg/s
Pressure drop	Δp	bar
Temperature difference	ΔT	°C

Variable	Symbol	Unit
Temperatures for first and last segment, by tube and gas	T_1/ T_N	°C
Electrical heat output	P_{el}	kW

HeatExchangerGas – heat exchanger gas to gas

Symbol



Function

The *HeatExchangerGas* component type is used to simulate a heat exchanger for the gases on the tube side and shell side. The simulation is implemented for these three types:

- Parallel-flow heat exchanger
- Counter-flow heat exchanger
- Cross-flow heat exchanger

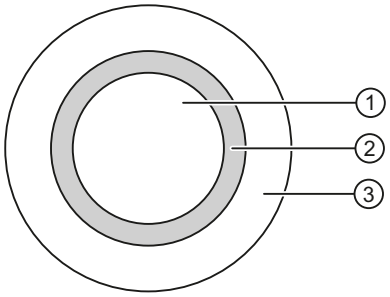
Both media are routed on the tube side via the connectors *TS_IN* and *TS_OUT*, and on the shell side via the connectors *SS_IN* and *SS_OUT*.

The flow $\dot{m}$ is throttled on the tube side and shell side according to the relationship

$$\frac{\Delta p}{\text{bar}} = \frac{-\dot{m}^2}{k_V^2 \, \rho_{TS} \frac{\text{kg}}{\text{m}^3}} 12900 \left(\frac{\text{sec}}{\text{h}} \right)^2$$

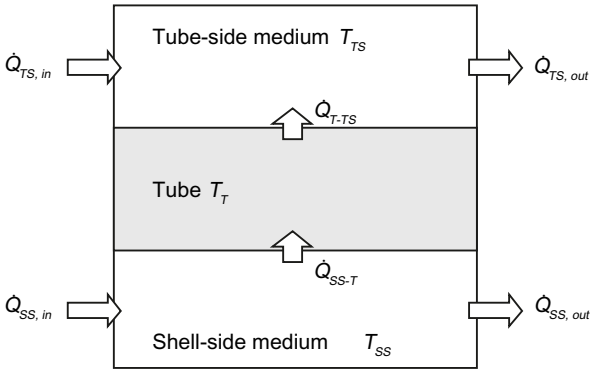
with the relevant throttle coefficient k_v. The chosen reference direction for the flow is from connector *_IN* to *_OUT* on the tube side and shell side.

For the heat transfer, a simple one tube model divided into segments is applied. The number *N* of segments can be set to a value between 4 and 16.



① Tube-side space
② Tube
③ Shell-side space

The gas heat on the tube and shell side and the heat in the tube itself are balanced.



The two heat transfers, from the shell side gas to the tube, and from the tube to the tube side gas are set according to

$$\dot{Q}_{SS-T} = A_{SS}\dot{q}_{SS-T} = A_{SS}\alpha_{SS}\left(T_{SS} - T_T\right)$$

$$\dot{Q}_{T-TS} = A_{TS}\dot{q}_{T-TS} = A_{TS}\alpha_{TS}\left(T_T - T_{TS}\right)$$

in an extremely simplified manner. For each segment *i* the heat balances

$$\frac{dT_{TS,i}}{dt} = a_{TS}\left(T_{TS,i-1} - T_{TS,i}\right) + b_{TS}\left(T_{T,i} - T_{TS,i}\right)$$

$$\frac{dT_{T,i}}{dt} = a_T\left(T_{SS,i} - T_{T,i}\right) + b_T\left(T_{TS,i} - T_{T,i}\right)$$

$$\frac{dh_{SS,i}}{dt} = a_{SS}\left(h_{SS,i-1} - h_{SS,i}\right) + b_{SS}\left(T_{T,i} - T_{SS,i}\right)$$

apply with the following coefficients, which are the same for all segments:

$$a_{TS} = \frac{N\dot{m}_{TS}}{\rho_{TS}V_{TS}}, \quad b_{TS} = \frac{A_{TS}\alpha_{TS}}{c_{TS}\rho_{TS}V_{TS}}, \quad a_{SS} = \frac{N\dot{m}_{SS}}{\rho_{SS}V_{SS}}, \quad b_{SS} = \frac{A_{SS}\alpha_{SS}}{c_{SS}\rho_{SS}V_{SS}}, \quad a_T = \frac{A_{SS}\alpha_{SS}}{M_T c_T}, \quad b_T = \frac{A_{TS}\alpha_{TS}}{M_T c_T}$$

The values set for the gases in the shell side and tube side of the flownet apply to the densities p_{TS}, ρ_{SS} and the specific heat capacities c_{TS}, c_{SS}.

For initialization, the temperatures of the tube segments are set to the temperature of the tube side gas, calculated from the specific enthalpy (*FNTS.HSPEC* input).

Parameters

Parameter name	Description	Unit	Default value
Type	Type of heat exchanger: *ParallelFlow*, *CounterFlow*, *CrossFlow*	–	*ParallelFlow*
NbrOfSegments	Number *N* of segments: 4 ≤ *N* ≤ 16	–	4
KvsSS	Shell side throttle coefficient k_V	m^3/h	360.0
KvsTS	Tube side throttle coefficient k_V	m^3/h	360.0
VolumeSS	Shell side volume V_{SS}	m^3	1.65
VolumeTS	Tube side volume V_{TS}	m^3	2.25
SurfaceSS	Shell-side surface A_{SS} of the tube (exterior surface of the tube)	m^2	720.0
SurfaceTS	Tube side surface A_{TS} of the tube (interior surface of the tube)	m^2	550.0
HeatTransCoefSS	Heat transfer coefficient α_{SS} on the tube exterior	kW/m^2K	0.1
HeatTransCoefTS	Heat transfer coefficient α_{TS} on the tube interior	kW/m^2K	0.1
sHeatCapTube	Specific heat capacity c_T of the tube	kJ/kgK	1.3
MassTube	Mass M_T of the tube	kg	1000.0

Operating window

The following variables are displayed in the operating window:

Variable	Symbol	Unit
Flow	$\dot{m}$	kg/s
Pressure drop	Δp	bar
Temperature difference	ΔT	°C
Temperatures for first and last segment, by • Tube • Gas (shell side) • Gas (tube side)	T_1/ T_N	°C

8.2.7 Creating your own component types for flownets

The CTE component type editor enables you to create your own component types, which utilize the mechanisms of the FLOWNET process. You can expand the functions of the components supplied with the FLOWNET library, for example, add physical effects that are not covered by the supplied library components and thus enhance your flownet simulations. You can also extend the FLOWNET library by creating your own component types from scratch.

Two aspects must be considered when creating components:

- The topological aspects
- The connection to the solution procedure

The topological aspect is covered by appropriate extensions to the component type definition. Special connection types are provided for the connection to the solution procedure.

In addition, the general descriptions of component properties in the "SIMIT – Component Type Editor" manual form the basis for creating flownet components. The general properties also apply in full to flownet components.

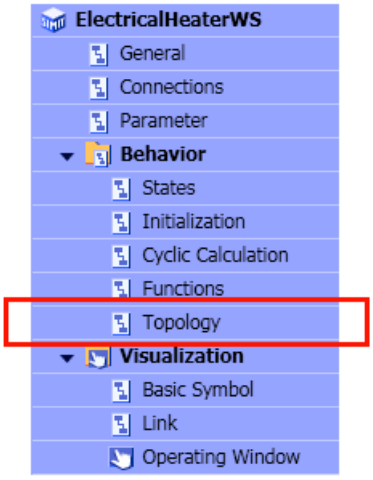
8.2.7.1 Topological properties

The topology of a flownet is automatically determined when compiling a simulation project from interconnected flownet components. Each flownet component must thus provide relevant topological information about itself, which means information about how it is to be treated topologically in the flownet. From a topological point of view it is necessary to know how the reference directions for variables in a flownet are defined for a component and how the data exchange between the component and the solution procedure for the flownet is set up.

The flownet elements with topological information are:

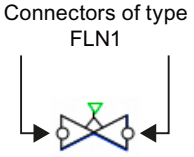
- Internal nodes
- External nodes
- Branch objects

Relationships with the topological connectors of the flownet component must be defined for each of these elements. To do this, open the topology editor by double-clicking on the *Topology* node in the navigation of the component type.

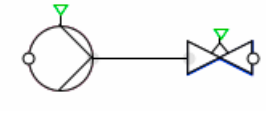


Connectors of the FLN1 connector type

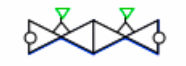
The *FLN1* connection type indicates the connectors of flownet components that are used to connect with one another. The topology of a flownet is derived from interconnected connectors of this type and the topological information on the individual flownet components. *FLN1* is therefore a purely topological connection type, which means it does not carry any signals. Connectors of this type are simply referred to below as **topological connectors**. Topological connectors are represented by circles.



If topological connectors are connected with each other by a connecting line in the chart editor, then the connected connectors are hidden. They are still visible only as pale shadows.



They can also be joined by superimposing the connectors. Once linked, both connectors are hidden.



Topology of the internal node

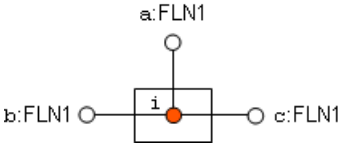
An internal flownet node is defined as follows in a component type in the topology description:

```
INTERNAL_NODE i;
```

Topological connectors of the component must also be assigned to this internal node. Three connectors can be assigned in the topological description as per the example below:

```
FROM i TO a;
FROM i TO b;
FROM i TO c;
```

The three connectors `a`, `b` and `c` must be defined as type *FLN1* connectors in the component type. The figure below shows a diagram of the resulting topological structure of the component type.



Topology of the external node

External nodes represent the boundaries of the flownet. External nodes can be used to predefine pressures and enthalpies in nodes and inflows and outflows for the flownet.

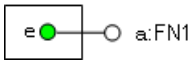
In the topological description of a component type, an external node e is defined as follows:

```
EXTERNAL_NODE e;
```

This external node must be also be linked to at least one topological connector of the component. The topology description is, for example, to be completed as

```
FROM e TO a;
```

. The topological connector `a` must be defined in the component type as a connector of type *FLN1*. The figure below shows a diagram of the resulting topological structure of the component type.



Topology of a branch object

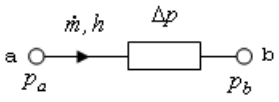
Branch objects are part of a flownet branch, such as throttling of pressure.

The reference direction for the variables of a branch object is defined in the topological description of the flownet component. For example, the definition

```
FROM a TO b;
```

defines the branch object with a reference direction from connector `a` to connector `b` of the component. Both of the topological connectors `a` and `b` must be defined in the component type as connectors of type *FLN1*.

Flow variables are positive in the reference direction. For a branch object, these are the mass flow $\dot{m}$ and the enthalpy flow $H = \dot{m}h$.



Reference direction

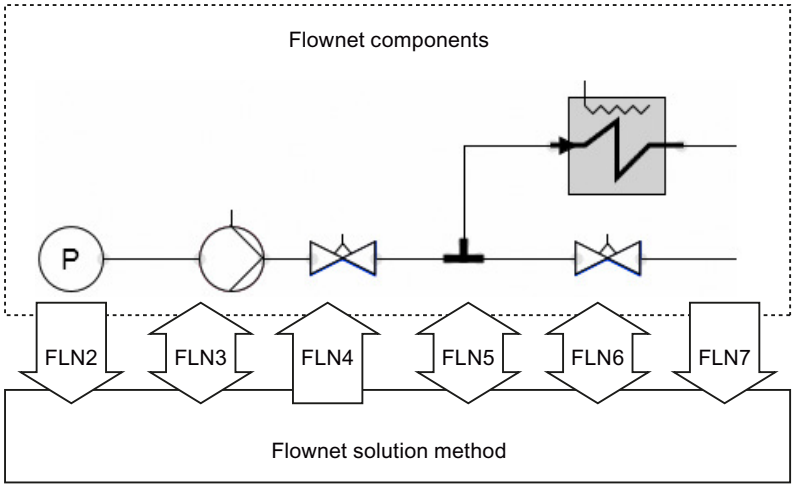
Pressure boosts are applied in the reference direction, which means the pressure change is defined here as

$\Delta p = p_b - p_a$

. For pressure drops in the reference direction (throttling), Δp is therefore negative and positive Δp for pressure boosts in the reference direction.

8.2.7.2 Connection to the solution procedure

The flownet objects of a component and the flownet solution procedure exchange data via special flownet connectors. There are six different connector types *FLN2* to *FLN7* available for this purpose. Their use in flownet componetns is explained in the sections Connector type FLN2 for branch objects (Page 856) to Connection type FLN7 for parameter assignment of an internal node (Page 861).



Connectors of the types *FNL2* to *FNL7* provide a connection to the flownet solution procedure and therefore must be set as **invisible** in the component symbol. In the connector properties, the usage "Only in property view" or "Only in CTE" must be set.

Property	Value
Usage	Property view only
Visibility Default	

See also

Connector type FLN6 for parameter assignment of a branch (Page 861)

Connector type FLN2 for branch objects

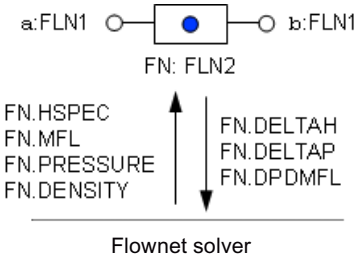
A branch object can exchange variables with the flownet solution procedure via a connector of type *FLN2*. The topological description of a branch object is completed as follows for a connector with the name *FN*:

```
FROM a TO b : FN;
```

The connector must always be defined in the *OUT* direction. It connects the component to the flownet solution procedure to exchange various variables that are relevant to the branch object between the flownet solution procedure and the component. It maps the branch-influencing effect (which means the effect that the branch object has on the branch) to the flownet.

Name	Connection type	Direction	Dimension
FNTS	FLN2	OUT	1

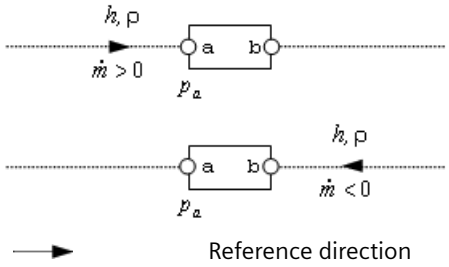
The figure below shows the input and output signals with the direction of data flow between the component and the flownet solution procedure.



The component receives four variables for calculating the effect of the branch object via the input signals:

1. Specific enthalpy *h* (*FN.HSPEC*),
2. Mass flow $\dot{m}$ (*FN.MFL*)
3. Pressure p_a (*FN.PRESSURE*),
4. Density ρ (*FN.DENSITY*).

The pressure p_a relates to the "from" connector (*FROM*) of the topological description, which is the connector `a` in our example. Density ρ and enthalpy *h* are variables of the supplied medium.



The output signals for the branch object are calculated in the component and sent to the flownet solution procedure:

1. Change in specific enthalpy Δ*h* (*FN.DELTAH*),
2. Pressure change $\Delta p = p_b - p_a$ (*FN.DELTAP*)
3. Derivation of the change in pressure based on the mass flow $d\Delta p/d\dot{M}$ (*FN.DPDMFL*).

Connector type FLN3 for external nodes

Variables are exchanged between external nodes and the flownet solution procedure via connectors of type *FLN3*. A connector of type *FLN3* is assigned to the external node in the topology description, as follows for a connector with name *FN*:

```
EXTERNAL_NODE e : FN;
```

The connector type *FLN3* provides signals in the forward and reverse direction as listed in the table below.

Table 8-41 Signals of the connector type FLN3

Forward signals		**Backward signals**	
Name	**Variable**	**Name**	**Variable**
HFW	Specific enthalpy	HBW	Specific enthalpy
PRESSURE	Pressure	MFL	Mass flow rate

The following applies: If the *FN* connector is defined with the direction *OUT*, then the pressure and specific enthalpy are specified via this connector for the external node of the flownet. If the connector is defined with the direction *IN*, the mass flow rate and the specific enthalpy are specified for the external node.

Connector of type FLN3 with direction OUT

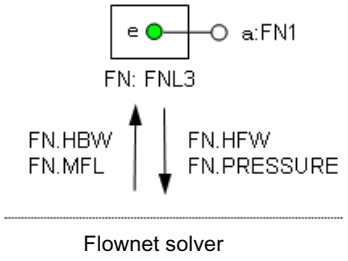
If the connector is defined with direction *OUT*, then the pressure calculated in the component is set for the flownet solution procedure via this connector. This means the external node is a **pressure node**. The following variables can be set at its outputs:

1. Pressure p_e (*FN.PRESSURE*),
2. Specific enthalpy h_e (*FN.HFW*).

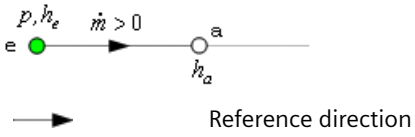
At its inputs, the mass flow supplied to or discharged from the flownet via the external node and calculated by the flownet solution procedure is returned to the components. In the case of discharge from the flownet, the specific enthalpy of the flow is also returned to the component:

1. Mass flow $\dot{m}$ (*FN.MFL*)
2. Specific enthalpy h_a (*FN.HBW*)

The figure below illustrates the signal direction for data exchange between the component and the flownet solution procedure.



Regardless of the direction selected in the topology description, the mass flow $\dot{m}$ is always positive for inflows to the flownet ($\dot{m} > 0$) and negative ($\dot{m} > 0$) for outflows.



Connector of type FLN3 with direction IN

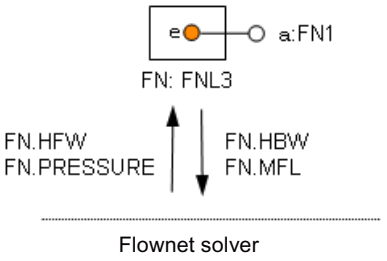
If the connector *FN* is defined with direction *IN*, then the mass flow set in the component is defined for the flownet via this connector. The external node active in the flownet is therefore a node with mass inflow or a mass flow node. The following variables can be set at its outputs:

1. Mass flow $\dot{m}$ (FN.MFL)
2. Specific enthalpy *h* (FN.HBW)

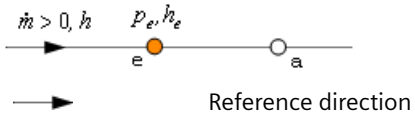
At its inputs, the pressure calculated by the flownet solution procedure for the external node is returned to the components. If discharge from the flownet is set with negative mass flow, then the specific enthalpy of the medium is returned to the component:

1. Pressure p_e (FN.PRESSURE)
2. Specific enthalpy h_e (FN.HFW)

The figure below illustrates the signal direction for data exchange between the component and the flownet solution procedure.



Mass flow nodes operate in the flownet like internal nodes with mass inflow or outflow. These nodes are therefore treated as internal nodes in the flownet, which means an internal node with additional inflow or outflow is created in the flownet for an external node. The pressure returned to the component p_e therefore corresponds to the pressure in this internal node, and the specific enthalpy h_e corresponds to the specific enthalpy of this internal node.



Connector type FLN4 for internal nodes

The variables for pressure, specific enthalpy and density calculated by the flownet solution procedure for an internal node can be used in a component via a connector of type *FLN4*. The topological description of the component type is completed as follows for a connector with the name *FN*:

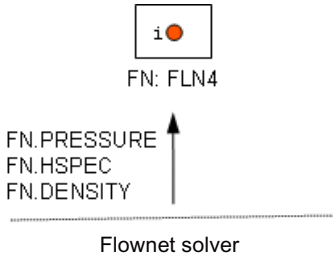
```
INTERNAL_NODE i : FN;
```

The connector must always be defined in the **IN direction**. It connects the component to the flownet to map the variables that are relevant for the internal node *i* from the flownet solution procedure to the component.

Name	Connection type	Direction	Dimension
FN	FLN4	IN	1

The variables calculated in the node *i* are provided by means of the connector inputs:

1. Pressure p_i (FN.PRESSURE)
2. Specific enthalpy h_i (FN.HSPEC)
3. Density ρ_i (FN.DENSITY)



Connector type FLN5 for parameters of a flownet

Parameters of a flownet can be exchanged between a component and the flownet solution procedure via a connector of type *FLN5*. This connector can be assigned to any flownet object: a branch object or an internal or external node. The topological definition of the object should be completed, for a connector with name *FN* as per

- `FROM a TO b : FN;`
- `INTERNAL_NODE i : FN;`
- `EXTERNAL_NODE e : FN;`

The signals of connector type FLN5 are listed in the table below. You can find their meaning in the mathematical model of the flownet in the following section: Parameter assignment of flownets (Page 627).

Note

The relationships used for calculation of the flownet variables, such as temperature, are defined via the *MEDIUM* parameter for the relevant medium. This can only be done during initialization of the simulation. Changes made to the parameters while the simulation is running have no effect.

Table 8-42 Signals of connector type FLN5

Signal	Meaning
MEDIUM	Flownet medium: MEDIUM := 0 for water/steam, MEDIUM := 1 for ideal gas, MEDIUM := 2 for liquid; must not be changed while the simulation is running
CG	Specific compression modulus *K* / *M* for water/steam medium with density ρ < 500 kg/m³ and for ideal gas medium
CL	Specific compression modulus *K* / *M* for water/steam medium with density ρ ≥ 500 kg/m³ and for liquid medium
MG	Thermal factor *1* / *M* for water/steam medium with density ρ < 500 kg/m³ and for ideal gas medium
ML	Thermal factor *1* / *M* for water/steam medium with density ρ ≥ 500 kg/m³ and for liquid medium
P_INIT	Initial value for the pressure in the internal nodes of the flownet
H_INIT	Initial value for the specific enthalpy in the internal nodes of the flownet
DENSITY	Density of the medium in the flownet when liquid is set as the medium
T_ENV	Ambient temperature
C_ENV	Heat transfer factor $c = \alpha A$ for heat exchange in the internal nodes of the flownet with the environment
L_CR	Specific heat capacity c_p for liquids as medium
IG_R	Specific gas constant R_S for ideal gas as medium
IG_CR	Specific heat capacity c_p for ideal gas as medium
ST	Binary signal; if ST := "True", then a linear transfer is applied for parameters CL, CG and ML, MG for the water/steam medium
AL	Momentum factor A for branches in the flownet

Depending on the direction of the connector *FN* its signals are inputs or outputs. Depending on the direction, flownet parameters can be set or used in component. If the connector is defined with direction *OUT*, the variables set in the component are provided to the flownet as parameters. These parameters are available in the component for evaluation, if the connector is defined with direction *IN*.

Connector type FLN6 for parameter assignment of a branch

The dynamics of the flow rate can be assigned parameters individually for each branch. To do this, a connector *FN*, via which the momentum factor A is transferred to the flownet solution procedure, must be added to the topological definition of the branch object as follows:

```
FROM a TO b: FN;
```

If a connector of type *FLN6* is defined with direction *OUT*, then the momentum factor A for the branch containing the object is transferred to the flownet solution procedure via this connector.

Name	Connection type	Direction	Dimension
FN	FLN6	OUT	1

If the connector is defined with direction *IN*, then the momentum factor A for the branch is transferred to the branch object by the flownet solution procedure.

Connector type FLN7 for parameter assignment of an internal node

Each internal node can be assigned parameters individually. A connector via which the parameters are transferred to the flownet solution procedure must be added to the topological definition of an internal node. For a connector *FN* the definition is completed as follows:

```
Internal_Node i: FN;
```

FN is always created as **an output** of type *FLN7* .

Name	Connection type	Direction	Dimension
FN	FLN7	OUT	1

The variables listed in the table below are transferred to the flownet solution procedure for the internal node. You can find their meaning in the mathematical model of the flownet in the following section: Parameter assignment of flownets (Page 627).

Table 8-43 Signals of connector type FLN7

Signal	Meaning
CG	Specific compression modulus K_i / M_i for water/steam medium with density ρ < 500 kg/m³ and for ideal gas medium
CL	Specific compression module K_i / M_i for water/steam medium with density ρ ≥ 500 kg/m³ and for liquid medium
MG	Thermal factor *1* / M_i for water/steam medium with density ρ < 500 kg/m³ and for ideal gas medium
ML	Thermal factor *1* / M_i for water/steam medium with density ρ ≥ 500 kg/m³ and for liquid medium

Signal	Meaning
P_INIT	Initial pressure value
H_INIT	Initial value for specific enthalpy
T_ENV	Ambient temperature
C_ENV	Heat transfer factor $c_i = \alpha A$ for heat exchange with the environment
L_CR	Specific heat capacity c_p for liquids as medium
IG_R	Specific gas constant R_S for ideal gas as medium
IG_CR	Specific heat capacity c_p for ideal gas as medium

8.2.7.3 Constants and functions

You can use various constants and functions when creating your own flownet components.

Constants

The available constants are listed in the table below.

Table 8-44 Constants for flownet components

Name	Data type	Value	Description
_GRAVITY	analog	9.81	Gravitational constant (gravity)
_T0	analog	273.15	Zero point temperature

Functions

Functions for calculating state variables are available for flownet components with water/steam medium. The available state variables and units are summarized in the table below.

Table 8-45 State variables for water/steam

Variable		Unit
p	Pressure	bar
p_S	Saturated steam pressure	bar
T	Temperature	°C
T_S	Saturated steam temperature	°C
h	Specific enthalpy	kJ/kg
h'	Specific enthalpy of saturated water	kJ/kg
h''	Specific enthalpy of saturated steam	kJ/kg
ρ	Density	kg/m³
ρ'	Density of saturated water	kg/m³
ρ"	Density of saturated steam	kg/m³

All state functions *FNAME* return a state variable *ZVAL* as an analog value. Calls are performed as follows:

ZVAL = _WaterSteam.*FNAME*(*PARAM1* (*, PARAM2*)).

The available state functions are described in the table below.

Table 8-46 State functions for water/steam

ZVAL	FNAME	PARAM1	PARAM2	State function
ρ	rph	p	h	$\rho = \rho(p,h)$
T	trh	ρ	h	$T = T(\rho,h)$
T_S	tvp	p_S	–	$T_S = T(p_S)$
p_S	pvt	T_S	–	$p_S = p(T_S)$
ρ'	rsvt	T_S	–	$\rho' = \rho'(T_S)$
ρ"	rssvt	T_S	–	$\rho'' = \rho''(T_S)$
ρ'	rsvp	p_S	–	$\rho' = \rho'(p_S)$
ρ"	rssvp	p_S	–	$\rho'' = \rho''(p_s)$
h'	hsvt	T_S	–	$h' = h'(T_S)$
h''	hssvt	T_S	–	$h'' = h''(T_S)$
h'	hsvp	p_S	–	$h' = h'(p_S)$
h''	hssvp	p_S	–	$h'' = h''(p_S)$

The saturated steam pressure p_S and saturated steam temperature T_S are limited for the saturation functions as follows:

$0 \le T_S \le 373.94$ °C and 0.0065 bar $\le p_S \le 220.64$ bar.

The relevant limit is set for values outside this range.

The values for pressure p and specific enthalpy h are limited to the following ranges for the state function $\rho = \rho(p,h)$:

0.007 bar $\le p \le 490$ bar and 20 kJ/kg $\le h \le 3998$ kJ/kg

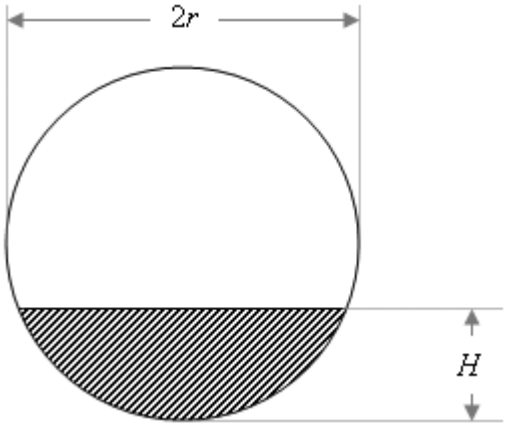
For the state function $T = T(\rho,h)$, the specific enthalpy is limited to the range

9 kJ/kg $\le h \le 4158$

kJ/kg. The limits for the density ρ depend on the enthalpy value: The lowest valid density value is 0.0012344 kg/m³ (with $h = 4160$ kJ/kg); the highest valid density is 1045.239 kg/m³ (with $h =$ 96 kJ/kg).

The auxiliary function *Cylniv* can be used to calculate fill levels in horizontal cylinders. For a cylinder of length L and radius r with specified volume V, it calculates a fill level of H. Its call is:

H = _Utilities.*Cylniv*(V, L, r).



User-defined functions

The FLOWNET library provides you with three different media: water/steam, ideal gas and liquid. The medium to be used is set in the signal *MEDIUM* of the connection type *FLN5* with a corresponding ID (0, 1, or 2). You can find additional information on this in section: Connector type FLN5 for parameters of a flownet (Page 860).

The state equations for density and temperature that are defined for the relevant medium are accordingly applied by the flownet solution procedure.

With the FLOWNET library you also have the option of performing flownet simulations with other media besides the pre-defined media. You have to provide the media-specific state equations for the calculation of density and temperature in the nodes by means of a “global function”.

Such a global function must consist of a .NET assembly that is named *Userdefined.dll* and which has its "Assembly Name" specified as "Userdefined" in the project settings. It must have the namespace *Userdefined* and a public class named *FlownetFunctions*. This class must contain two public static functions for calculation of the state variables:

- *public static double trh(long m, double r, double h)*
- *public static double rph(long m, double p, double h)*

The function *trh* has to calculate the temperature from the density *r* and the specific enthalpy *h*; the function *rph* has to calculate the density from the pressure *p* and the specific enthalpy *h*. The units of the variables are to be assumed as follows:

- Temperature in °C,
- Density in kg/m^3
- Pressure in bar, and
- Specific enthalpy in kJ/kg.

The medium has to be specified by a freely selected negative number. This negative media key number forms the first transfer parameter *m* of both functions. In your own functions you only need to consider the user-defined negative media key numbers; if the value given is ≥ 0 computation will be performed by SIMIT, and your functions will not be called in this case.

This means you have the option of defining any desired media in these two functions and of using them in a simulation project. Note that in your user-defined functions, you only need to pay attention to negative media key numbers. For non-negative media key numbers, the functions that are included in SIMIT are used. Your user-defined functions are not called by SIMIT in this case.

The scope for your user-defined functions might be set up as follows:

```
namespace Userdefined
{
 public class FlownetFunctions
 {
  public static double trh(long m, double r, double h)
  {
   switch (m)
   {
    case -1:
     // Calculation for media with key number -1 …
     return 0.0;
    case -2:
     // Calculation for media with key number -2 …
     return 0.0;
    default:
     // Should not occur!
     return 0.0;
   }
  }

  public static double rph(long m, double p, double h)
  {
   switch (m)
   {
    case -1:
     // Calculation for media with key number -1 …
     return 0.0;
    case -2:
     // Calculation for media with key number -2 …
     return 0.0;
    default:
     // Should not occur!
     return 0.0;
   }
  }
 }
}
```

To assign a specific medium to a flownet, set the relevant media key number in the *MEDIUM* signal of the *FLN5* connector type; in the case of user-defined media this means setting the relevant negative value.

Note

If you set a negative key number for the medium without providing the necessary functions for the calculation of density and temperature, the flownet solution procedure assumes that the density and temperature are zero.

You can call these two functions using

_FlownetFunctions.trh(*m*, *r*, *p*)

or

_FlownetFunctions.rph(*m, p, h*)

even in your own component types. Moreover you can implement additional functions in the class *FlownetFunctions* and use them by means of a corresponding function call in your component types.

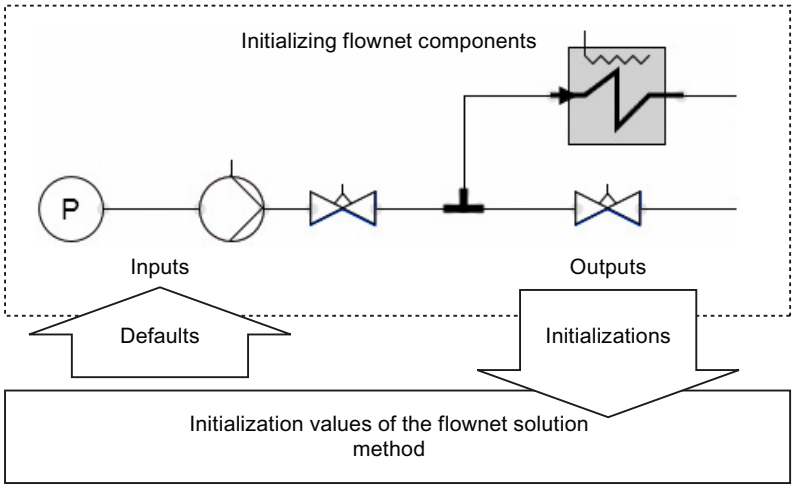
You must store the assembly in the SIMIT workspace, in the subfolder *GlobalFunctions*. The assembly will then automatically be copied to each new project and will therefore be archived with your project.

Note

Modifications to the assembly do not affect existing simulation projects. The modified assembly affects newly-created projects. If you want to use the modified assembly in projects that already exist, copy it into the *globalFunctions* folder in these projects.

8.2.7.4 Initializing FLOWNET simulations

The initialization of flownet simulations occurs in two steps. When the simulation is started up, first the components are initialized, which means the simulation model created from the connected components is initialized. Then the flownet solution procedure is initialized. The flownet solution procedure uses the values that are initialized in the components and passed to it via connectors of type FLN2 through FLN7.



Since no variables of the flow network solution are available for component initialization, it is advisable to preassign suitable values to the component type inputs. We recommend values that are consistent with the default values of the flownet solution procedure.

8.3 CONTEC library

8.3.1 Introduction

The *CONTEC* library is an extension of SIMIT, which provides component types for creating simulations of conveyor systems. By linking components of this library, you can create a model of a conveyor system in SIMIT to simulate the transport of objects in this system. The *CONTEC* library provides a special solution procedure for this purpose for use in SIMIT. Once the simulation has started, the solution procedure continuously calculates the positions of the objects in the modeled conveyor system.

The *CONTEC* library is only suitable for simulating conveyor systems for individual objects; bulk material conveyor systems cannot be simulated. In addition, the simulation is limited to conveyor systems in which the transportation routes for the objects are fixed by the arrangement of the handling equipment. Conveyor systems with autonomously operating vehicles, which means vehicles that determine their own transportation route, are therefore excluded.

SIMIT provides a real-time simulation for the virtual commissioning of user programs for automation systems. Likewise, the simulation of conveyor systems with the CONTEC library is always used in a closed circuit with real or simulated automation systems. This sort of simulation with SIMIT is therefore fundamentally different from the material flow simulations used for

planning and designing conveyor systems. There, the material flow in a conveyor system is simulated on the basis of a pre-defined control strategy, whereas here the control strategy is translated into an actual automation system, which includes a simulation of the purely mechanical elements of the conveyor system created with SIMIT. To differentiate it, this type of simulation is therefore referred to as material handling simulation in this manual.

As is usual in SIMIT, material handling simulations are easy to create with the *CONTEC* library, using components from the graphical user interface. The emphasis in implementing the component types in the library was on the simple creation and parameter assignment of the simulation model and on the stability of the model in the simulation, rather than on a detailed simulation of the mechanical aspects of conveyor systems. Material handling simulation is therefore based on a movement model that simulates the movement of objects along paths at a pre-defined speed, the paths being determined by the handling equipment. The weight of the object and other factors influencing movement, such as friction, for example, are disregarded.

Two types of conveyor systems can be simulated with the component types provided in the *CONTEC* library:

- Conveyor systems with rail-mounted vehicles
  such as electrical overhead monorails, ground transport systems, etc. and
- Non-vehicular conveyor systems
  such as chain, roller and belt conveyor systems, etc.

Component types are also available in *CONTEC* that allow you to include RFID, Moby, bar code, etc. in the simulation.

You can use SIMIT to create your own library components for material handling simulations and thus expand your material handling library. You can use the special *CONTEC* solution procedure for material handling simulation in the component types by means of special connection types.

Note

If there are material handling components in your simulation project, which means components that use the material handling solution procedure, then the simulation can only be run if you have a material handling license.

8.3.2 Material handling simulation

Material handling describes the technology used for moving objects in any direction over limited distances. Objects can generally be moved in any spatial direction. Handling equipment refers to the individual machines that are used to move the object in the conveyor system.

The *CONTEC* material handling library can be used to create simulation models of a conveyor system and to run simulations with them. The handling equipment is subject to the following restrictions:

- All transportation routes are defined as linear paths.
- Transportation routes and possible branches must be predefined as a structure of the conveyor system for a simulation.

In addition, the simulation of objects is restricted as follows:

- Only individual items can be considered as objects, not bulk goods.
- All objects are represented by a box-shaped outline.
- Other than its dimensions, no other physical properties of the object (for example weight, friction, etc.) are taken into consideration.

The component types contained in the *CONTEC* library can be divided into two categories:

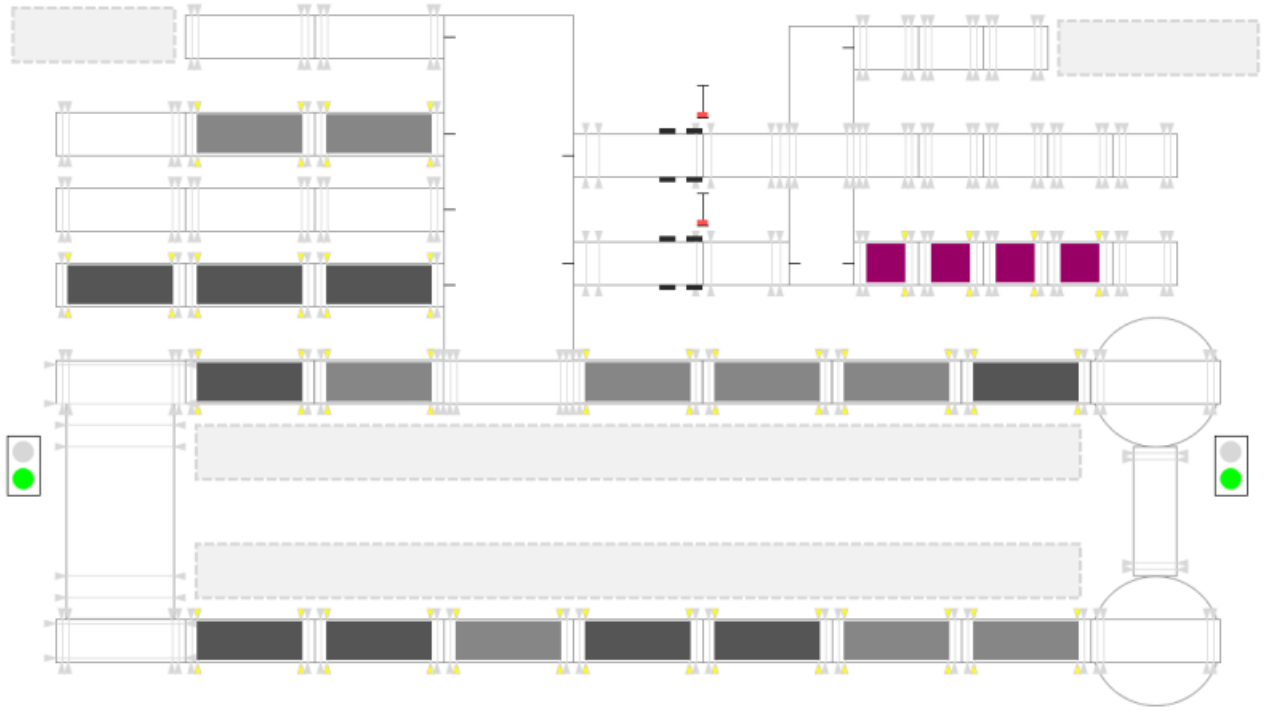
- Object component types and
- Equipment component types

As is usual in SIMIT, equipment component types are added to charts as components using their symbol. The symbols for these component types are designed so that when put them together they create a scaled layout of the conveyor system, as shown for example in the figure below. The resulting system model then only needs to be provided with appropriate parameters and linked to the automation.

Along with the system model, the objects defined for a simulation project are stored in table form. There are symbols for object component types, too, with which the individual objects can be displayed correctly in the conveyor system layout when the simulation is running.

Material handling simulation is based on a special solution procedure that is configured and programmed by means of the individual components of the simulation model. This solution procedure is based on a movement model which allows the position of the objects on the paths defined by the handling equipment to be determined at any moment in the simulation. The solution procedure takes account of the fact that objects can be held up, and it uses the position and dimensions of the objects to activate sensors in the simulation.

The movement of an object on a path can be initiated in the simulation by both the equipment components and the object components. In this way it is possible to simulate both conveyor systems such as roller conveyor systems, in which the drive acts on the equipment, and rail-mounted conveyor systems, in which the drive acts on the vehicle, which means the object.



8.3.2.1 Principles of conveyor technology simulation

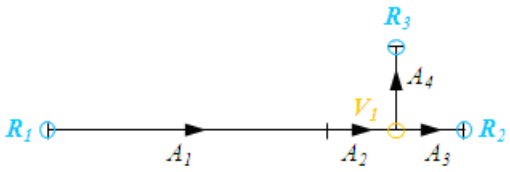
The material handling simulation method is based on a movement model that assumes that the objects move along linear paths, the possible paths being determined by the handling equipment. The way in which the handling equipment is configured defines a conveyor system network that represents the possible transportation routes for the objects in a conveyor system.

Complex motions can be realized by self-defined components and connected to the system in linear pathways. You can find additional information on this in section: Connector type MT8 for transferring objects at boundary points (Page 954).

A network consists of individual sections and branches. It can be inherently closed or open. In an inherently closed network both ends of each section connect to a branch, whereas in an open network at least one end of one section is not connected to a branch. This forms a boundary point of the network. The smallest possible conveyor system network consists of one section.

A network section is made up of one or more segments. The geometry of a segment can consist of either a straight path of a given length or a circular arc of a given radius and angle. In addition, every segment has a direction. The direction of a segment is the reference direction for the movement of the object over this segment. It is defined in such a way that a positive speed value moves the object in the reference direction, while a negative speed value moves the object against the reference direction.

By way of example, the figure below shows a conveyor system network consisting of three sections or four segments A_1 to A_4 and one branch V_1.



This section is clearly an open conveyor system network with the three boundary points R_1, R_2 and R_3. If the object crosses the boundary of a conveyor system network it is lost, which means it is removed from the handling equipment and returned to the material list where it is once more available for use. Components can remove objects from or feed them into the conveyor system network at the boundaries by means of suitable connections to the solution procedure.

The special solution procedure of the material handling calculates the position of the objects at equidistant times, as is usual in SIMIT. It supplements the standard SIMIT solution procedure so that material handling component types can be used alongside other component types, for example, component types from the basic library. To enable data to be exchanged, material handling components are connected to the material handling solution procedure by means of specific connections. They receive values calculated by the solution procedure and send variables to the solution procedure via these connections.

For each segment in the conveyor system network, positions can be defined at which sensors detect the object as it passes. The material handling solution procedure then ensures that the relevant sensor signals are activated in the simulation according to the positions and dimensions of the objects.

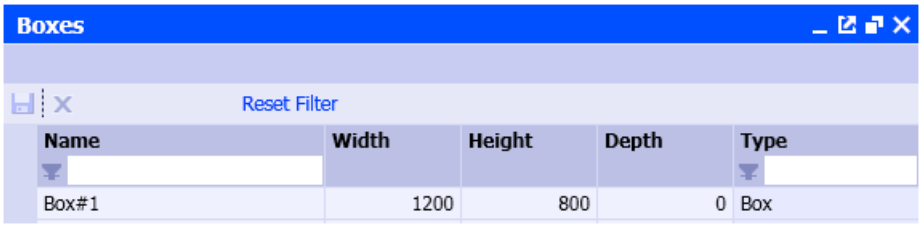
8.3.2.2 Modeling of the objects

Material transport objects are modeled in SIMIT as component types. For each type, at least the size and the graphical representation of the object must be defined. Connectors, parameters, states and behaviors can also be defined for an object in its component type.

The link symbol is used where possible to represent the object in charts: if a link symbol is defined, this is used for the display; otherwise the basic symbol is used, with any connectors on the symbol being hidden.

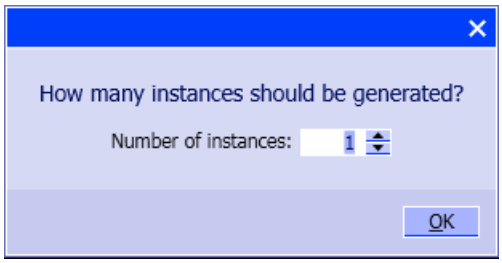
To simulate objects the available stock of object components has to be defined in the SIMIT project. For this, select "New list" node in the project view under the "Lists" project folder.This creates a material list, which means a list of the object components available in the simulation.

The editor for creating the material list opens:



Here you can define which object components are available in the simulation. For each component type you can specify the number of instances to be formed. To create an instance, simply drag the component type you want from the *MATERIAL* directory of the *CONTEC* library into the editor. To create more than one instance, hold down the Alt key as you drag the

component type. A query dialog will appear in which you can specify the number of instances, as shown in the figure below:



You can also create several material lists in the "Lists" project folder and sort the material lists into different folders if necessary.

In order to place object components on a specific section of the simulated transport network for the start of the simulation, you can assign one or more object components of the same type from the material lists in the project to the component simulating the section.

You can find examples for this in the following sections:

- Rail-S4 – Straight rail with four sensors (Page 888)
- Conveyor-S4 – Straight conveyor with four sensors (Page 905)

If object components have a behavior model (behavior description), this is executed at the start of the simulation in the same way as it would be if the object components were placed on a chart like other components. The simulation of the behavior of an object defined by that description is entirely independent of the treatment of that object in the solution procedure for the material handling simulation.

Once the simulation has started, each object defined in the material list is given a unique identification number which is automatically set by SIMIT and displayed in the material lists.

Boxes

Reset Filter

Obj-ID	Name	Width	Height	Depth	Type
1	Box#1	1200	800	0	Box
2	Box#2	1200	800	0	Box
3	Box#3	1200	800	0	Box

If an object is detected at a sensor during the simulation, the solution procedure returns the ID of the detected object component.

8.3.2.3 Modeling the conveyor system network

Conveyor system networks are made up of segments, branches and boundaries. The *CONTEC* library provides various component types for modeling the conveyor system network. These component types represent elementary handling equipment types and therefore encapsulate corresponding structures consisting of segments, branches and boundaries. Thus the complexity of these subnetworks modeled in the component types can differ. For example, in a component type representing a straight conveyor only a single segment is modeled. By contrast, component types representing more complex handling equipment, like a 90-degree transfer, for example, have a model made up of multiple segments and branches.

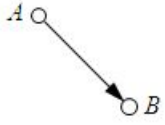
The individual component types have connectors representing end points of segments or branch points. If the connectors of two components of these types are connected on charts, the subnetwork models of the two components are connected accordingly. By connecting them to other components the complete conveyor system network of a conveyor system is ultimately constructed.

For every segment the geometry (with coordinates) and the reference direction have to be specified. Optionally, the positions of stoppers and sensors may also additionally be defined for a segment. For each branch it must be defined which segments it connects, and these must be provided with corresponding information about their positions in the conveyor system network.

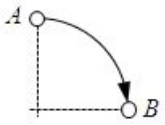
Modeling segments

A segment is modeled as a line. It is defined by its two end points, the geometry and the direction of the segment. The geometry can take the form of a straight line or an arc. It is defined by the position of its two end points *A* and *B* and its angle. If the angle is not zero, there are two possible segments which can be differentiated by the sign of the angle: a positive angle means that the center point of the arc is located to the right of the segment in the counting direction, while a negative angle means that the center point of the arc is located to the left of the segment. The radius of the arc is determined by these three variables. The direction of the segment corresponds to the reference direction for transporting the objects.

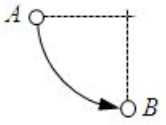
Angle 0



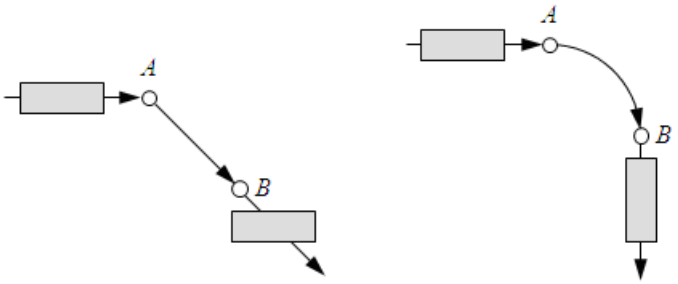
Angle +90



Angle –90



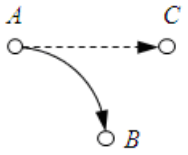
The object being transported over a segment is rotated by the angle of the segment. This means on straight segments the object is transported without being rotated; the absolute orientation of the object is maintained. The figure below shows the transport of an object over a straight and a curved segment:



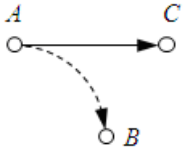
For each segment a speed is defined at which the objects are transported over that segment. Positive speed values cause the object to be transported in the reference direction, while negative speed values cause it to be transported against the reference direction. The inherent speed of an object is added to the segment speed where applicable.

A state is assigned to each segment for movement in the reference direction and for movement against the reference direction. This state identifies a segment as "active" or "inactive" in the corresponding direction. Inactive segments are regarded as non-existent in the material handling solution procedure. This means that objects cannot be moved on these segments, and the positions of objects already located on these segments are not updated. In addition, sensors on inactive segments are not updated. The change in state of segments is used to specify an explicit transportation route at branches. The figure below shows an example of how a switch function can be modeled by alternately activating and deactivating the two segments in the reference direction:

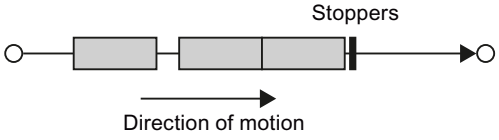
Segment *AC* inactive



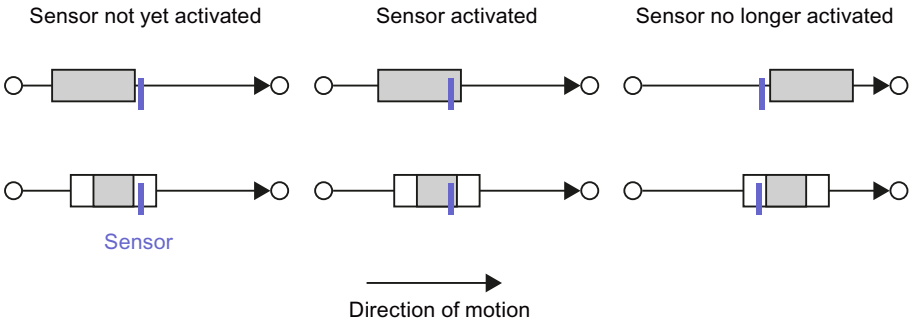
Segment *AB* inactive



Positions at which the transport of an object can be blocked can be defined on a segment. The state of a stopper position can be set to "active" or "inactive". If a stopper is active, all objects are stopped at this position and all downstream objects are held up accordingly as shown in the figure below by way of example:



In addition, positions at which an object can be detected can be defined on a segment as sensor positions. At each of these sensor positions the identifier (ID) of the object covering the position is then recorded in the material handling solution procedure. The detection range for the objects can also be defined for each sensor position. This range is defined as a percentage of the object size and is assumed to be symmetrical with respect to the center of the object. The figure below illustrates the detection of an object for two different detection ranges: the top row shows the activation of a sensor for the full detection range (100%) of the object; the bottom row shows the activation of a sensor for half the detection range (50%). The detection range for the object is shown in gray in each case and movement is assumed to be from left to right.

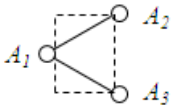
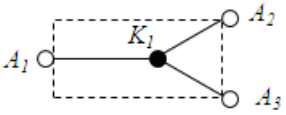


Component types in which only one segment with sensors is modeled include, for example, the straight rail (*Rail-S4*) and the straight conveyor (*Conveyor-S4*).

Modeling branches

Because all sections are assumed to be linear, a branch is a point in the conveyor system network at which more than two segments connect.

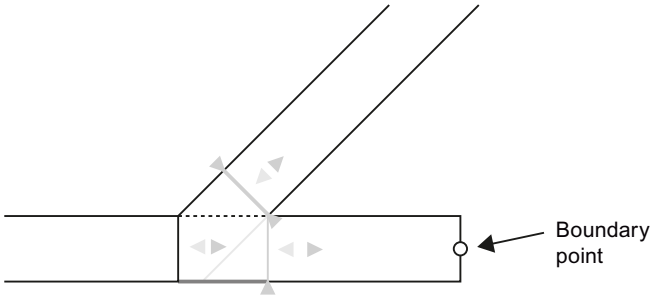
In the subnetwork of a component type a branch can either be shown as a connector, which means a shared end point of multiple segments (as shown in figure below on the left), or it can be located within a subnetwork (as shown in figure below on the right). In the first case the position of the branch point is given directly by the position of the connector, whereas in the second case the branching position is defined relative to the component.

(a) Branch at a connector	(b) Branch in a component
	

At branch points the transportation route must be explicitly defined at every moment of the simulation. In other words, no more than two of the segments that connect at a branch may be active as a transportation route at any given time. Therefore activation of the segments must be included as a function in the component type.

Modeling boundaries

Boundaries are end points of a segment that are not connected to other segments. For a component like a straight conveyor, for example, a boundary of the conveyor system network is created if a connector of this component is not connected to other components. In the simulation, if an object moves beyond such a boundary point it is removed from the conveyor section and returned to the material list. A message to that effect appears in the message line.



Boundary points can also be defined as end points in the subnetwork of component types. The behavior of an object passing this boundary point can then be freely defined in the component type using suitable functions. For example, the position of the object can be freely calculated to simulate complex movements. You can find examples for such components in the following sections:

- Turntable-R60 – Turntable (Page 928)
- TransferCarriage – Transfer carriage (Page 933)

8.3.2.4 Special features of conveyor technology simulation

Placing and removing objects

Objects that are defined in the available material stock for the simulation project can be placed on any segment in the simulation. They are placed at the start of the segment in accordance with the specified reference direction (FROM-TO). Objects are always positioned relative to their geometric center.

Objects can also be removed from the network. There is a corresponding system call for placing and removing objects, which allows this function to be simulated in component types.

Simulating the holdup behavior

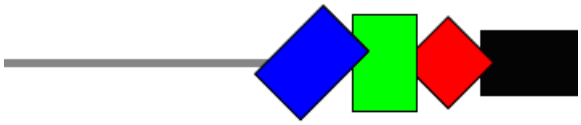
Holdups on straight conveyor sections

Modeling of the holdup behavior has been simplified to a great extent. This means the positions of held-up objects in the simulation do not necessarily correspond to the actual positions in all cases.

The holdup behavior on straight sections is correct for all objects that are either not rotated at all or are rotated in multiples of 90°.



Objects with other rotation angles ϕ are also held up. However, as their position can only be calculated approximately, this leads to overlaps.



The effective width $\tilde{b}$ of an object that is used as a simplification to simulate the holdup is calculated using the following formula:

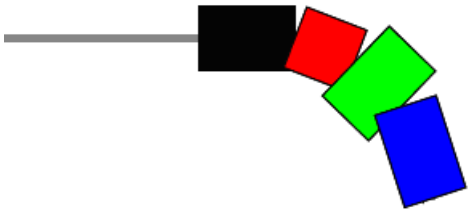
$$\tilde{b} = \frac{\varphi h + (\pi - 2\varphi) b}{\pi}$$

where *h* denotes the height and *b* the width of the object.

Holdup in trends

Modeling of the holdup behavior has been simplified to a great extent. This means the positions of held-up objects in the simulation do not necessarily correspond to the actual positions in all cases.

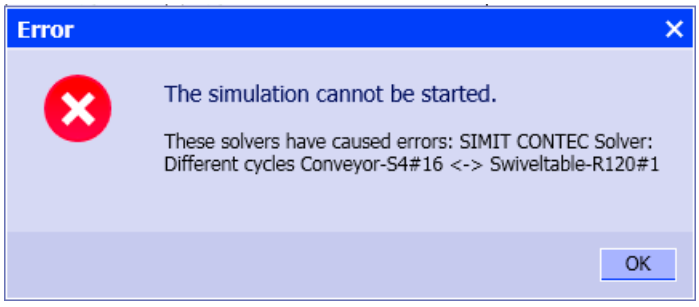
In trends, too, the position of held-up objects is only an approximation. The held-up objects overlap, as shown in the figure below.



The overlaps are caused by the fact that the bend of the trend is disregarded in the distance calculation.

Time slice assignment for components

All components forming a cohesive conveyor system network must be assigned to the same time slice. This means all components that simulate handling equipment and are directly connected to one another via connectors or signal lines must be assigned to the same time slice. Otherwise the simulation model of a cohesive conveyor system network will be formed from multiple sub-models with differing cycle times. The simulation start request will then be denied and a corresponding error message displayed.

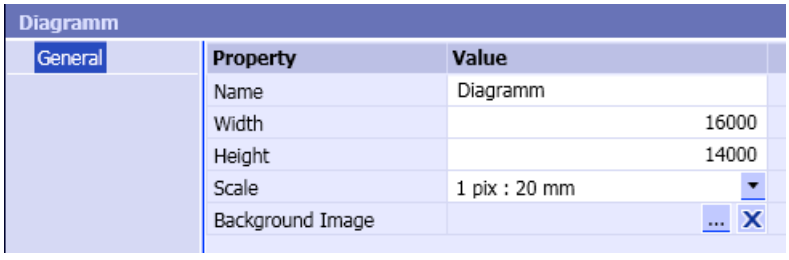


8.3.2.5 Scalability

Unlike in the standard library, the dimensions of components play a decisive role in the conveyor technology library because they directly indicate the length of conveyor sections or the dimensions of objects. In accordance with standard engineering practice, millimeters [mm] are used throughout as the length unit, while speeds are given in meters per second [m/s].

Scalability of conveyor sections

To enable conveyor systems of differing sizes to be mapped on charts with a reasonable resolution, a scale can be assigned to each chart.



The following scales are possible:

1 pix : 1 mm,

1 pix : 5 mm,

1 pix : 10 mm,

1 pix : 20 mm,

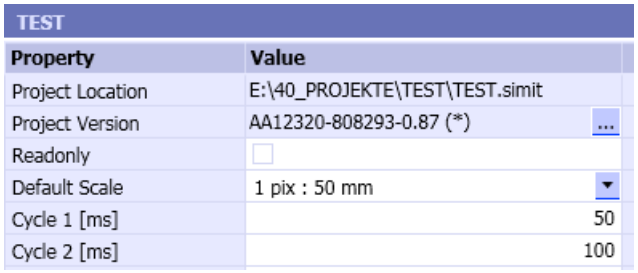
1 pix : 50 mm,

1 pix : 100 mm.

Note

If you have licensed the material transport library for your SIMIT installation, the sizes of all components, including components from the basic library, are reproduced at the specified scale. For basic library components you can retain the intended sizes by choosing a scale of 1 pix : 1 mm for charts containing basic components.

To avoid having to set the scale individually for each chart, you can choose a default setting for the entire project in the property view of the Project Manager.



This scale is used as the default for all new charts that you create.

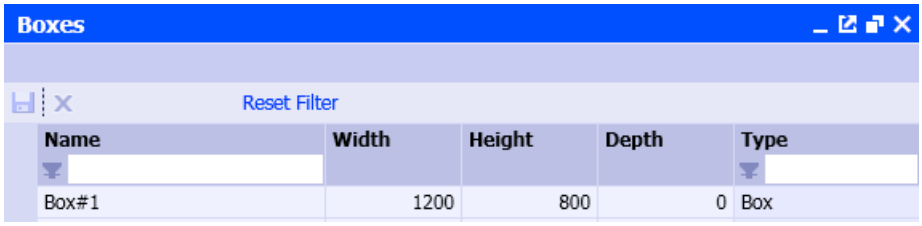
Note

Setting a default scale in the Project manager does not change the scale of any existing charts.

Note that changing the scale of a chart does not change the displayed size of a component, but rather its effective dimensions in millimeters.

Scalability of objects

Objects are shown in the material lists of a SIMIT project with their absolute size (width and height) in millimeters. This means objects are represented on charts in differing sizes, depending on the chosen scale of the chart.



The dimensions of an object are pre-defined in the component type. So when you add an object to the material list, it is added with the width, height and depth defined by the component type.

You can then change the dimensions for each object in the material list, provided that the component is scalable. The object types provided in the CONTEC library are all scalable.

Note

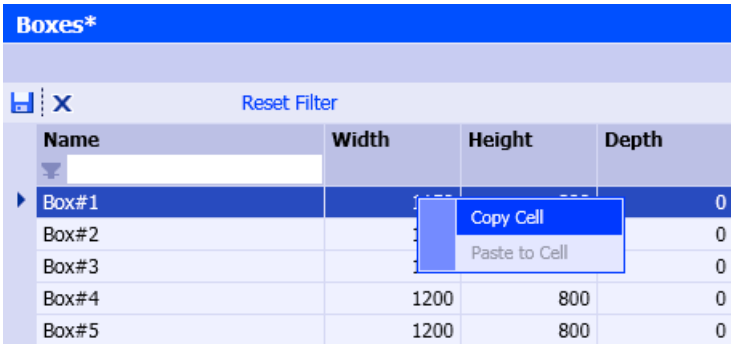
Note that *width* and *height* refer to the dimensions of the object as viewed from above, as shown in the diagram. The *depth* is the size in the third dimension, which is not shown. This definition was chosen so as to retain the meaning of the height and width of components and graphics on a chart. The component types provided in the material handling library disregard the depth of the object. With the component types available in the library, objects of any depth are detected by sensors solely on the basis of their width and height. For that reason the depth of objects is set to zero by default. The depth of objects only becomes important if you create a custom material handling component type that evaluates the depth of an object, for example a height check, and use it in your material handling simulation.

You can find additional information on this in the section: Creating custom component types for material handling simulation (Page 944).

If you want to set the dimensions of several objects to the same value, you can copy the value for one object, in other words from one row of this table to any number of others. Follow the steps outlined below:

- Select the row containing the required object value.
- Right-click on the cell containing the value to be copied and select "Copy Cell" in the shortcut menu.

- Select the rows to which you want to copy the value.
- Right-click on the column to which you want to copy the value and select "Paste to Cell" in the shortcut menu.





8.3.2.6 Generating the simulation of drives and sensors

As conveyor sections can be shown to scale on charts as a system layout using appropriate components, there is no point in also showing all additional input and output signals for these components in graphical form in these charts. This would compromise the clarity of the system layout. This means the only visible connectors on the component types in the *CONTEC* library are those used to connect them to each other to construct the system layout. Connectors for connecting components to the associated drive and sensor signals, and hence via the couplings to the controller signals, are implemented by means of an implicit assignment of the inputs. You have various options for connecting components. They are explained below by reference to a simple rail.

Manual connection of components

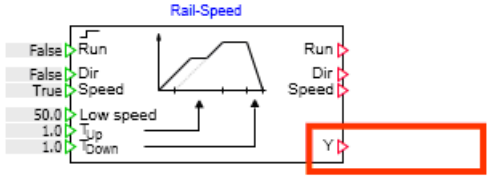
The component type of a rail is created in such a way that the component receives its percentage speed value as an input value from output *Y* of another component whose name is formed from the name of the rail and the suffix *Speed*. For example, the figure below shows a rail with the name *Rail*.



This means its *Speed* input is predefined so that it is implicitly connected to output *Y* of the component with the name *Rail-Speed*.



You can therefore create a chart which in this example contains a component with the name *Rail-Speed* and has an output *Y* which then defines the percentage speed for this rail.



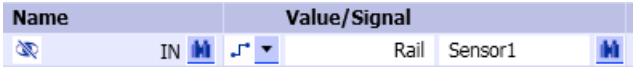
Alternatively, you can also enter every other (analog) output of a component in the *Speed* input for the rail by changing the setting from {$} to and entering the corresponding output signal, as shown by way of example in the figure below.



If you want the speed defined by the rail to be constant, you can also set the *Speed* input to 123 and enter the desired percentage directly.



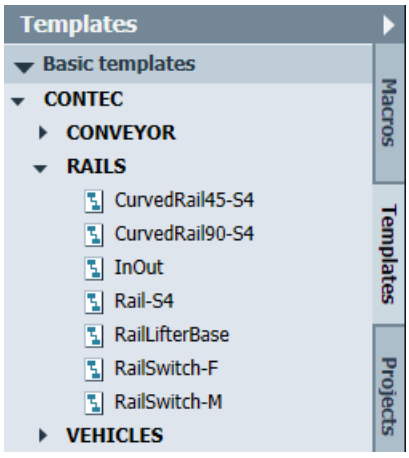
If you want to be able to further process output signals for the rail sensors, you would likewise use implicit connections. You can use any components and assign their (binary) inputs accordingly, as shown by way of example in the figure below.



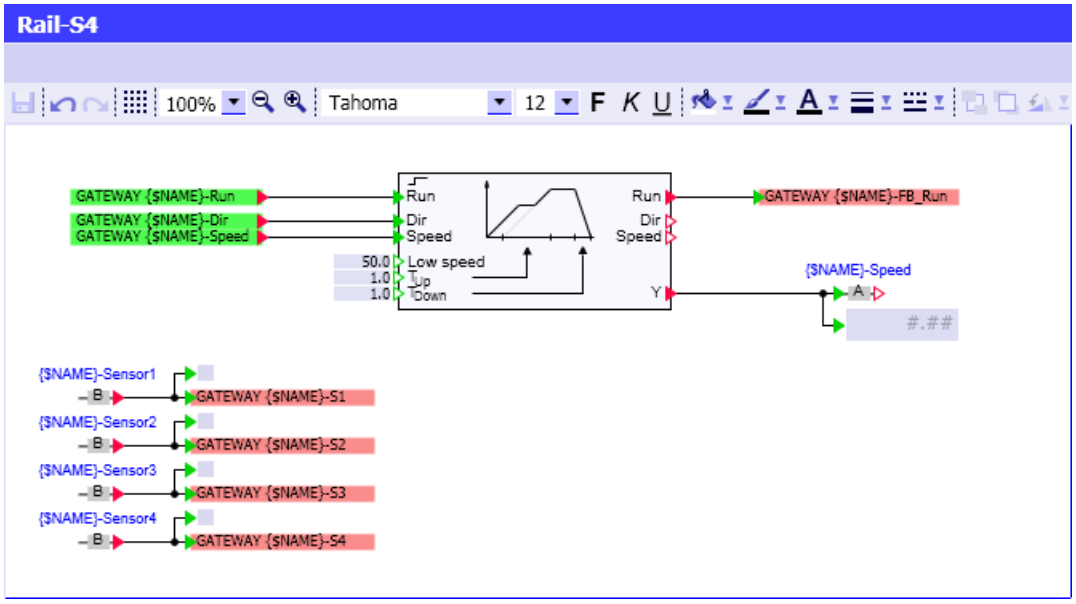
Using templates

Templates can be used as a way of creating charts for drive and sensor simulation with minimal effort.

The material handling library includes suitable templates for connecting the drive and sensor signals for every component type of a conveyor section, as well as for the vehicles. The template names are the same as the names of the component types.



All templates include a component from the basic SIMIT library for simulating the drive. In addition, the template can also be used to connect the simulated sensors in the conveyor section to the controller via coupling signals. The template for the "Rail-S4" component type is shown in the figure below.



If you use an identification system that allows the symbolic names of the I/O addresses to be derived from the names of the conveyor sections, you can modify these templates accordingly. You can then obtain the full connection of the corresponding conveyor section to the controller via the coupling by instantiating the template.

To connect all conveyor sections in the simulation project to the controller with the minimum possible effort, use the "Instantiate templates" function.

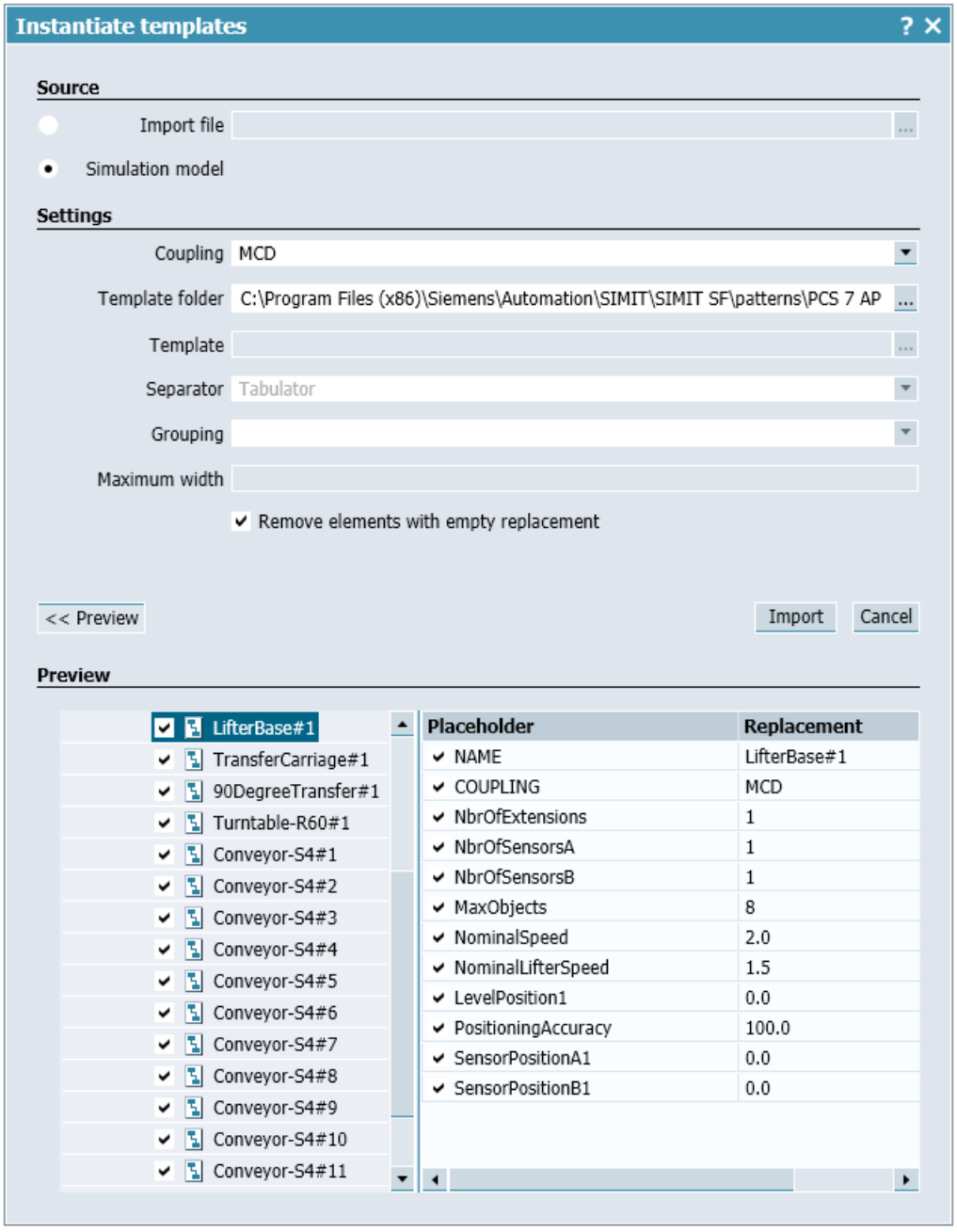
You have the following options for calling this function:

- In the portal view, select "Automatic model creation" > "Instantiate templates".
- From the Project view, select the menu command "Automatic model creation > Instantiate templates".
- Select "Automatic model creation" > "Instantiate templates" from the shortcut menu of a chart folder.

You can use the "Instantiate templates" > "Simulation model" function to create an instance of a template for every component in your simulation project that has a *TEMPLATE* parameter. The name of the template to be instantiated is determined using this *TEMPLATE* parameter. Using its *HIERARCHY* parameter the component can also define a folder hierarchy in which the template is instantiated.

The components of the *CONTEC* library are designed for use with this method. They contain the (additional) parameters *TEMPLATE* and *HIERARCHY*. If you have used CONTEC components in your SIMIT project, the "Instantiate templates" > "Simulation model" function provides the associated charts for connecting the controller.

As a reference is usually made to the coupling signals in the templates for generating the device level, you can specify a coupling name in the import dialog. It can be entered under the variable name *GATEWAY*. If your simulation project already includes couplings, they are listed in the drop-down list.



Note

You have to adapt the names of the input/output signals in the templates to your identification system to gain the maximum benefit from this method.

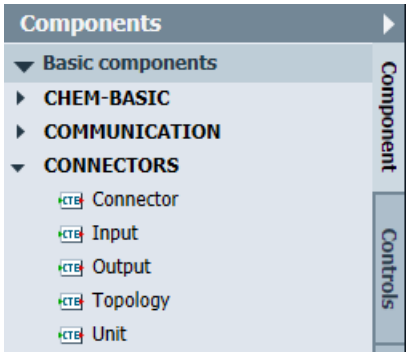
The minimum size for a template is 20 × 20 pixels.

8.3.3 Components of the CONTEC library

8.3.3.1 Topological connector in the CONTEC library

The *CONNECTORS* directory of the SIMIT basic library includes a connector that can be used to create topological connectors for material handling components that transcend chart limits:

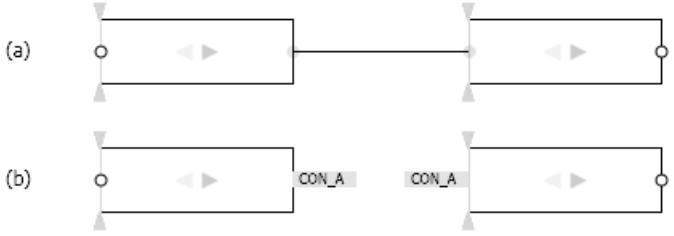
- the *Topology* connector.



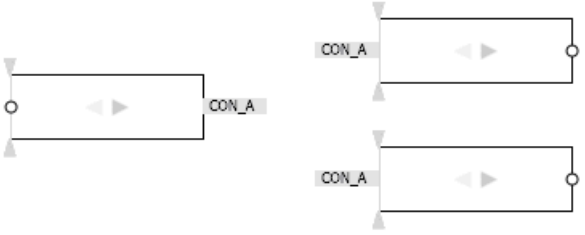
The *Topology* symbol is shown in the figure below:



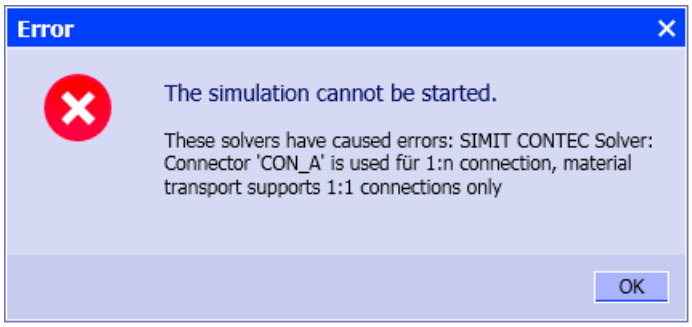
Using the *Topology* connector, a topological connection can be created between two or more material handling components. Under (b) in the figure below, two components are connected by the *CON_A* connector. The connection is functionally identical to the direct connection of both components via a connecting line, as shown under (a) in the figure below.



The figure below shows three components connected by the *CON_A* connector. This configuration is not valid, because with material handling components only two topological connectors can ever be connected to one another.



If you start a simulation project containing a configuration like this, an error message will appear as shown in the figure below and the simulation is aborted.



Note

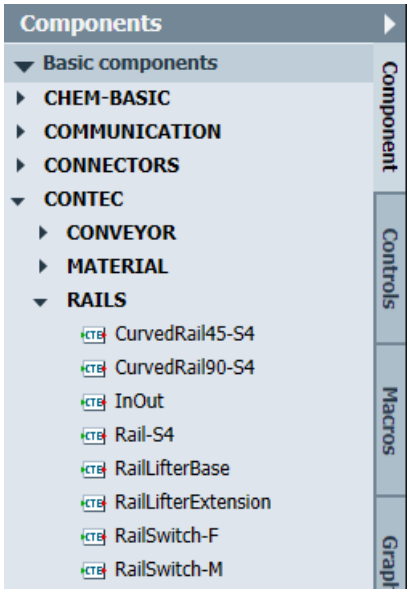
Note that cross-chart connections to the following components cannot be directly realized via connectors. Instead, at least one of the CONTEC components not listed here must be interposed:

- *Turntable-R60*
- *LifterBase*
- *90DegreeTransfer*
- *Swiveltable-R120*
- *TransferCarriage*
- *RailLifterBase*

The *ConveyorAdapter* or *RailAdapter* components can be used for this purpose in particular.

8.3.3.2 Component types for conveyor systems with vehicles

The *RAILS* directory of the CONTEC library contains component types for use in vehicular conveyor system simulations. In the broadest sense these types simulate "rails" on which vehicles can be placed as objects. The vehicles are either moved at the speed set by the individual rail segment or the vehicles set the speed themselves. A superposition of both speeds is also possible, although this is not used in practice.



Rail-S4 – Straight rail with four sensors

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the vehicles. The width of the symbol is scalable. The width corresponds to the length of the rail section.

Function

The *Rail-S4* component type is used for simulating a straight rail section. The speed at which vehicles are moved over this rail section is set at the hidden *Speed* input of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter).

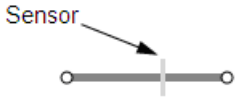
Between one and four sensors can be positioned along the length of the rail. When a vehicle is within the detection range of a sensor, the corresponding hidden binary output *Sensor1* to *Sensor4* is set. The ID of the detected vehicle can be obtained via the corresponding hidden integer outputs *SensorId1* to *SensorId4*.

Parameters

The behavior of the component can be set by means of parameters:

- *NominalSpeed*
  Nominal conveyor speed; adjustable online
- *NbrOfSensors*
  Number of sensors used (1 to 4)
- *SensorPosition*
  Position of the corresponding sensor relative to connector *A* of the symbol.

When the simulation is running, every defined sensor is shown in the correct position in the symbol. When a sensor is activated, its color changes to yellow.



The parameters, their units and default values are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0
NbrOfSensors		1
▾ SensorPosition [1]		...
SensorPositi...	[mm]	8000.0

The validity of the parameters is checked while the component is being initialized. The following messages indicate possible parameter assignment errors:

- *Parameter 'SensorPosition' must be less than the component's width*
  The sensor position must not be outside the component.
- *Parameter 'SensorPosition' must not be negative*
  The sensor position must not be negative.

Additional parameters

The *Rail-S4* component includes the additional parameters *TEMPLATE* and *HIERARCHY*, which can be used to control generation of the drive level. You can find more information in section: Using templates (Page 882).

Further additional parameters can be used during initialization to place vehicles on the rail.

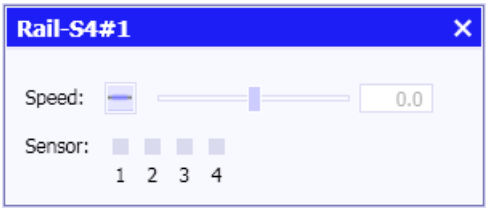
- *MaterialType*
  The type of vehicle that is to be placed
- *MaterialList*
  The name of the material list from which the vehicle is to be taken. If this parameter is left blank, all the material lists in the simulation project are searched.

- *InitNbrOfObjects*
  The initial number of vehicles to be placed on the rail
- *Clearance*
  The distance between the vehicles that are initially placed

Name		Value
MaterialType		
MaterialList		
InitNbrOfObjects		0
Clearance	[mm]	0.0
TEMPLATE		Rail-S4
HIERARCHY		RAILS

Operating window

In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider and monitor the status of a maximum of four sensors.



CurvedRail45-S4 – 45° curved rail with four sensors

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the vehicles. The size of the symbol can be scaled proportionally. The size corresponds to the size of the rail section. Proportional scaling alters the radius of the curve; the angle of 45° does not change.

Function

The *CurvedRail45-S4* component type is used to simulate a curved rail with a 45° bend. The speed at which vehicles are moved over this rail section is set at the hidden *Speed* input of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter).

Between one and four sensors can be positioned along the length of the rail. When a vehicle is within the detection range of a sensor, the corresponding hidden binary output *Sensor1* to

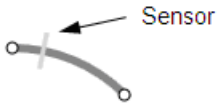
Sensor4 is set. The ID of the vehicles can be obtained via the hidden integer outputs *SensorId1* to *SensorId4*.

Parameters

The behavior of the component can be set by means of a parameter:

- *NominalSpeed*
  Nominal conveyor speed; adjustable online
- *NbrOfSensors*
  Number of sensors used (1 to 4)
- *SensorPosition*
  Position of the corresponding sensor relative to connector *A* of the symbol.

When the simulation is running, every defined sensor is shown in the correct position in the symbol. When a sensor is activated, its color changes to yellow.



The parameters, their units and default values are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0
NbrOfSensors		1
▼ SensorPosition [1]		...
SensorPositi...	[mm]	8000.0

The validity of the parameters is checked while the component is being initialized. The following messages indicate possible parameter assignment errors:

- *Parameter 'SensorPosition' must be less than the component's length*
  The sensor position must not be outside the component.
- *Parameter 'SensorPosition' must not be negative*
  The sensor position must not be negative.

Additional parameters

The *CurvedRail45-S4* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

Name	Value
TEMPLATE	CurvedRail45-S4
HIERARCHY	RAILS

Operating window

In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider and monitor the status of a maximum of four sensors.



CurvedRail90-S4 – 90° curved rail with four sensors

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the vehicles. The size of the symbol can be scaled proportionally. The size corresponds to the size of the rail section. Proportional scaling alters the radius of the curve; the angle of 90° does not change.

Function

The *CurvedRail90-S4* component type is used to simulate a curved rail with a 90° bend. The speed at which vehicles are moved over this rail section is set at the hidden *Speed* input of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter).

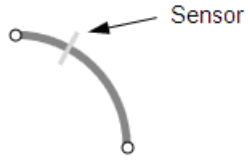
Between one and four sensors can be positioned along the length of the rail. When a vehicle is within the detection range of a sensor, the corresponding hidden binary output *Sensor1* to *Sensor4* is set. The ID of the vehicles can be obtained via the hidden integer outputs *SensorId1* to *SensorId4*.

Parameters

The behavior of the component can be set by means of parameters:

- *MaxSpeed*
  Nominal conveyor speed; adjustable online
- *NbrOfSensors*
  Number of sensors used (1 to 4)
- *SensorPosition*
  Position of the corresponding sensor relative to connector *A* of the symbol.

When the simulation is running, every defined sensor is shown in the correct position in the symbol. When a sensor is activated, its color changes to yellow.



The parameters, their units and default values are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0
NbrOfSensors		1
▾ SensorPosition [1]		...
SensorPositi...	[mm]	8000.0

The validity of the parameters is checked while the component is being initialized. The following messages indicate possible parameter assignment errors:

- *Parameter 'SensorPosition' must be less than the component's length*
  The sensor position must not be outside the component.
- *Parameter 'SensorPosition' must not be negative*
  The sensor position must not be negative.

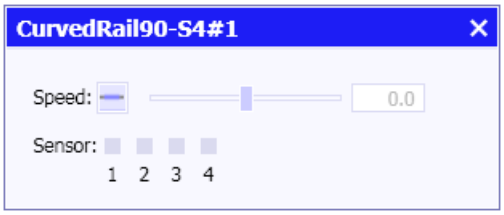
Additional parameters

The *CurvedRail90-S4* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

Name	Value
TEMPLATE	CurvedRail45-S4
HIERARCHY	RAILS

Operating window

In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider and monitor the status of a maximum of four sensors.



RailAdapter – Connection component for cross-chart connection

Symbol



Function

The *RailAdapter* component type is used for simulation a straight rail section. The speed at which vehicles are moved over this rail section is specified at the invisible *Speed* input of the component as a percentage value of the configurable nominal conveyor speed (*NominalSpeed* parameter).

This simple rail section can be used in particular to make cross-chart connections.

Parameter

The behavior of the component can be set by means of a parameter:

- *NominalSpeed*
  Nominal conveying speed in m/s; can be changed online.

Additional parameters

The *RailAdapter* component includes the additional parameters *TEMPLATE* and *HIERARCHY*, which can be used to control generation of the drive level. You can find more information on this in section: Using templates (Page 882).

Operating window

In the operating window, you can specify the conveyor speed as a percentage value of the nominal conveyor speed (*NominalSpeed*) using a slider.

RailSwitch-F – Switchable switch with 45° spur

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the vehicles. The size of the symbol can be scaled proportionally to adjust the size of the switch. Proportional scaling alters the length of the switch and the radius of the spur curve, but the spur angle of 45° does not change.

Function

The *RailSwitch-F* component type is used to simulate a branch (switch) or junction with a 45° spur. It functions as a switch when objects are moved in the reference direction and as a junction when they are moved against the reference direction.

If the hidden binary input *Switch* is set to one (True), transport is switched from the straight section (*A – B*) to the spur (*A – C*). Note that transport against the reference direction is likewise only possible over an active segment.

The speed at which vehicles are moved over the rails is set at the hidden *Speed* input of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter).

The switch cannot be operated if it is occupied by vehicles. If an attempt is made to operate an occupied switch, the following error message appears: "Switch cannot be operated while being occupied".

Parameters

The behavior of the component can be set by means of a parameter:

- *NominalSpeed*
  Nominal conveyor speed; adjustable online

The parameter, its unit and default value are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0
SwitchingTime	[s]	1.0

Additional parameters

The *RailSwitch-F* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

Name	Value
TEMPLATE	RailSwitch-F
HIERARCHY	RAILS

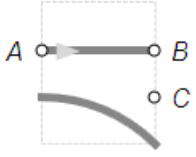
Operating window

In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider and switch the sections.



RailSwitch-M – Movable switch with 45° spur

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the vehicles. The size of the symbol can be scaled proportionally to adjust the size of the switch. Proportional scaling alters the length of the switch and the radius of the spur curve, but the spur angle of 45° does not change.

Function

The *RailSwitch-M* component type is used to simulate a branch (switch) or junction with a 45° spur. It functions as a switch when objects are moved in the reference direction and as a junction when they are moved against the reference direction.

The hidden analog input *Switch* is used to move the switch from transport over the straight section (*A – B*) to transport over the spur (*A – C*). The switching time is configurable (*SwitchingTime* parameter). Positive values at the Switch input move the switch to the *A – C* section, while negative values move it to the *A – B* section. The absolute values are percentages of the configurable switching speed. The moving operation for a component is shown in animated form in the symbol.

The speed at which vehicles are moved over the rails is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter).

The switch cannot be moved if it is occupied by vehicles. If an attempt is made to move an occupied switch, the following error message appears: "Switch cannot be operated while being occupied".

Parameters

The behavior of the component can be set by means of parameters:

- *NominalSpeed*
  Nominal conveyor speed; adjustable online
- *SwitchingTime*
  Switching time for the switch when triggered with +/- 100% values at the *Switch* input.

The parameters, their units and default values are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0
SwitchingTime	[s]	1.0

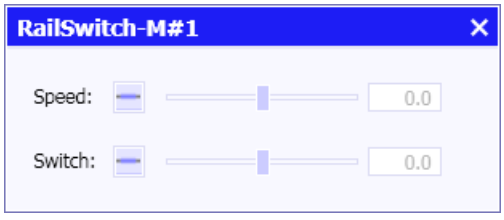
Additional parameters

The *RailSwitch-M* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

Name	Value
TEMPLATE	RailSwitch-M
HIERARCHY	RAILS

Operating window

In the operating window, you can use a slider to set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) and the switching time as a percentage of the configurable switching time (*SwitchingTime*).



InOut – Inward and outward section for vehicles

Symbol



The width of the symbol is scalable. The set width corresponds to the length of the inward/outward section.

Function

The *InOut* component type represents a section over which vehicles are moved into or out of a conveyor system network. The InOut component is connected to an open section of the network at connector *A* to extend the conveyor system network.

To move them into the network, individual vehicles are placed on the unconnected end of the inward/outward section (left edge of the symbol) by setting the hidden binary input *CreateObject* and moved from there to connector A. Correspondingly, vehicles to be moved out of the network are moved from connector A to the unconnected end of the inward/outward section, from where they are removed from the conveyor system network. The speed at which vehicles are moved over the inward/outward section is set at the hidden *Speed* input of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter).

The inward/outward section can only be occupied by one vehicle at a time. If the section is occupied by a vehicle, no other vehicle can be positioned to be moved into the network or removed from the network.

Parameters

The type of vehicles to be moved into the network, the material list from which they are taken and the speed at which the vehicles are moved over the inward/outward section can be set by means of parameters.

The parameters, their units and default values are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0
MaterialType		
MaterialList		

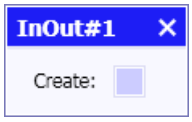
Additional parameters

The *InOut* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in the section: Using templates (Page 882).

Name	Value
TEMPLATE	InOut
HIERARCHY	RAILS

Operating window

In the operating window, you can place a new vehicle on the inward/outward section by clicking the pushbutton.



RailLifter – Lifter

A lifter is a rail that can be moved vertically through several levels. In SIMIT a lifter is made up of the base component of the *RailLifterBase* type and up to eight expansion components of the *RailLifterExtension* type. The base component simulates the lifter in the base level, while each additional level is simulated by an expansion component. A lifter that can be moved through up to eight additional levels can be simulated in this way.

You do not have to position the base component and the associated expansion components on the same chart. You can create a separate chart for each level, for example, and distribute the lifter components over the individual charts.

The following points apply to the parameter assignment of the base component:

- There are no restrictions on the name of the base component.
- The *NbrOfExtensions* parameter indicates the number of additional levels and hence also the number of expansion components used.
- In the base component you have to specify the position (*LevelPosition*) for each level, which means the level above the base level in millimeters. The base level is by definition at level zero. All level values must be positive and increase with the level number.

The following points should be borne in mind when assigning the parameters for the expansion components:

- The name of the expansion component is formed from the name of the base component, the '#' character, and a number indicating the level.
- The *Level* parameter must correspond to the level number. The numbering of the levels starts with one.
- The *BaseName* parameter is the name of the base component.
- The length of the lifter is determined by the width of the base component.

Note

All expansion components must be exactly the same width as the base component. Otherwise, a message will be displayed pointing to this inconsistency when the simulation is started.

The rail drive and the lifter sensors are fully implemented in the base component. This means expansion components do not work without the base component.

RailLifterBase – Lifter (base component)

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the vehicles. The width of the symbol is scalable. The set width corresponds to the length of the lifter rail.

Function

The *RailLifterBase* component type is used to simulate the base station of a lifter. The speed at which vehicles are moved over the lifter rail is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter). Similarly, the lifting speed, which means the speed at which the rail is moved to the different levels, is set at the hidden analog input *LifterSpeed* as a percentage of the configurable nominal lifting speed (*NominalLifterSpeed* parameter).

To move vehicles in and out at the various lifter levels, the lifter rail must be moved to the appropriate level and stop flush with that level. Flushness must be set with a positioning accuracy Δ that is the same for all levels (*PositioningAccuracy* parameter). The rail stops flush with a level *H* if the following applies for its level h:

$H - \Delta \leq h \leq H + \Delta$

Between one and four sensors can be positioned on the rail relative to each of the two connectors *A* and *B*. When a vehicle is within the detection range of a sensor, the corresponding hidden binary output *Sensor1* to *Sensor4* is set. The ID of the detected vehicle can be obtained via the corresponding hidden integer outputs *SensorId1* to *SensorId4*.

Note

The base component has further hidden inputs for exchanging signals with extension components. The corresponding signals are assigned to these inputs by default. Do not change these default settings, otherwise the lifter components will not behave in the intended way in the simulation.

Parameters

The behavior of the component is configurable.

- *NominalSpeed*
  Nominal conveyor speed; adjustable online
- *NominalLifterSpeed*
  Nominal lifting speed; adjustable online
- *NbrOfExtensions*
  Number of additional levels (1 to 8)
- *LevelPosition*
  Level of the corresponding additional level
- *PositioningAccuracy*
  Positioning accuracy within which the level is deemed to have been reached.
- *NbrOfSensorsA*
  Number of sensors relative to connector A (1 to 4)
- *SensorPositionA*
  Position of the corresponding sensor relative to connector *A* of the symbol
- *NbrOfSensorsB*
  Number of sensors relative to connector B (1 to 4)
- *SensorPositionB*
  Position of the corresponding sensor relative to connector *B* of the symbol

When the simulation is running, the defined sensors are shown in the correct position in the symbol. When a sensor is activated, its color changes to yellow.

The parameters, their units and default values are shown in the figure below.

Name	Value
NominalSpeed [m/s]	2.0
NominalLifterSp... [m/s]	1.5
NbrOfExtensions	1
▼ LevelPosition [1]	...
LevelPosition1 [mm]	0.0
PositioningAccu... [mm]	100.0
NbrOfSensorsA	1
▼ SensorPositionA [1]	...
SensorPositi... [mm]	0.0
NbrOfSensorsB	1
▼ SensorPositionB [1]	...
SensorPositi... [mm]	0.0

Additional parameters

The *RailLifterBase* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

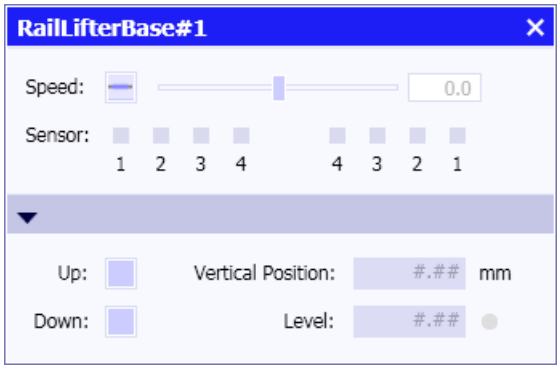
You also need to specify how many vehicles can be present on this conveyor section at any one time.

Name	Value
MaxObjects	8
TEMPLATE	RailLifterBase
HIERARCHY	RAILS

Operating window

In the operating window, you can use a slider to set the transport speed as a percentage of the nominal conveyor speed (*NominalSpeed*) and you can monitor the status of the sensors.

In the extended operating window you can move the lifter up or down by means of pushbuttons. The current lifter position and level are displayed.



RailLifterExtension – Lifter (extension component)

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the vehicles. The width of the symbol is scalable. The set width corresponds to the length of the lifter rail.

Note

The width of the extension component must be the same as the width of the base component.

Function

The *RailLifterExtension* component type is used to simulate a lifter at one of the accessible levels. A component of this type can only be used in combination with a component of the *RailLifterBase* type.

Parameters

The behavior of the component is configurable:

- *Level*
  Level at which the component is located. The numbering starts at one.
- *BaseName*
  Name of the corresponding base component

The parameters, their units and default values are shown in the figure below.

Name	Value
Level	1
BaseName	

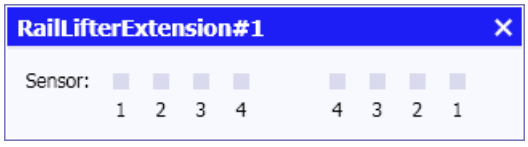
Additional parameters

Specify how many objects can be present on the section (lifter rail) defined by the component at any one time. This parameter should be set to the same value as for the base component.

Name	Value
MaxObjects	8

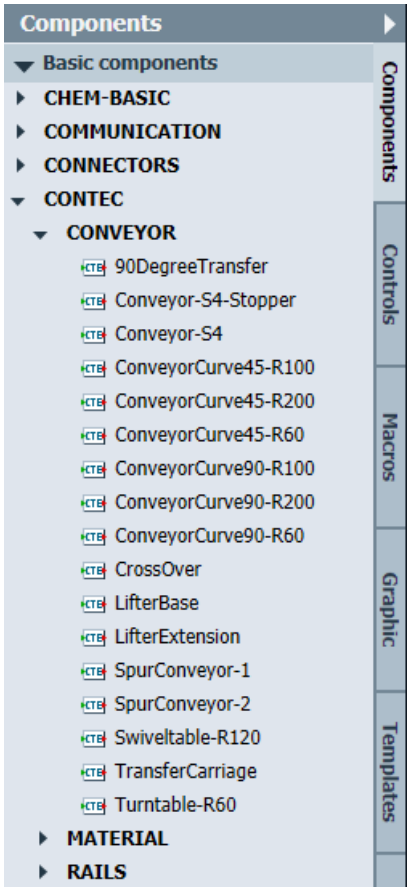
Operating window

In the operating window, you can monitor the status of the sensors.



8.3.3.3 Component types for non-vehicular conveyor systems

The *CONVEYOR* directory of the *CONTEC* library contains component types for use in simulating conveyor systems such as roller, chain and belt conveyor systems. All of these components determine the speed at which the object is moved. The object is passive in respect of these components, in other words it generally does not have its own drive. These components are typically used to transport pallets, containers or capital goods.



All component types of the *CONVEYOR* library have a fixed symbol height of 40 pixels. According to the selected scale the width of the conveyors can be adjusted to the values listed in the table below.

Table 8-47 Selectable widths for conveyors

Conveyor width	Scale
40 mm	1 pix: 1 mm
200 mm	1 pix: 5 mm
400 mm	1 pix: 10 mm
800 mm	1 pix: 20 mm
2000 mm	1 pix: 50 mm
4000 mm	1 pix: 100 mm

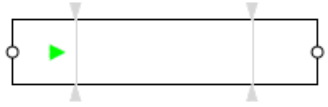
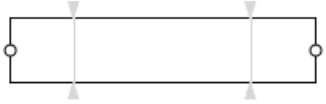
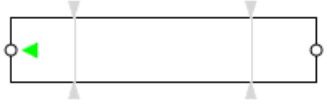
Conveyor-S4 – Straight conveyor with four sensors

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the objects. When the simulation is running the current transport direction is indicated by green arrows in the symbol.

Table 8-48 Indication of the current transport direction in the symbol

	Transport in the reference direction
	No transport
	Transport against the reference direction

The width of the symbol is scalable. The set width corresponds to the length of the conveyor.

Function

The *Conveyor-S4* component type simulates a straight conveyor section of a given length. The speed at which objects are moved over this conveyor is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter).

Between one and four sensors can be positioned on the conveyor relative to each of the two connectors *A* and *B*. When an object is within the detection range of a sensor, the corresponding hidden binary output *Sensor1* to *Sensor4* is set. The ID of the detected object can be obtained via the corresponding hidden integer outputs *SensorId1* to *SensorId4*.

Parameters

The behavior of the component can be set by means of parameters:

- *NominalSpeed*
  Nominal conveyor speed; adjustable online
- *NbrOfSensorsA*
  Number of sensors relative to connector *A* (1 to 4)

- *SensorPositionA*
  Position of the corresponding sensor relative to connector *A* of the symbol
- *NbrOfSensorsB*
  Number of sensors relative to connector *B* (1 to 4)
- *SensorPositionB*
  Position of the corresponding sensor relative to connector *B* of the symbol

When the simulation is running, the defined sensors are shown in the correct position in the symbol. When a sensor is activated, its color changes to yellow.

The parameters, their units and default values are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0
NbrOfSensorsA		1
▼ SensorPositionA [1]		...
SensorPositionA1	[mm]	0.0
NbrOfSensorsB		1
▼ SensorPositionB [1]		...
SensorPositionB1	[mm]	0.0

Additional parameters

The *Conveyor-S4* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

Further additional parameters can be used to place objects on the conveyor for the simulation. These are placed on the conveyor after the start of the simulation, starting at end *A*:

- *MaterialType*
  The type of object initially placed on the conveyor
- *MaterialList*
  The name of the material list from which the object is to be taken. If this parameter is left blank, all the material lists in the simulation project are searched.
- *InitNbrOfObjects*
  The number of objects that are to be placed
- *Clearance*
  The distance between the objects that are to be placed

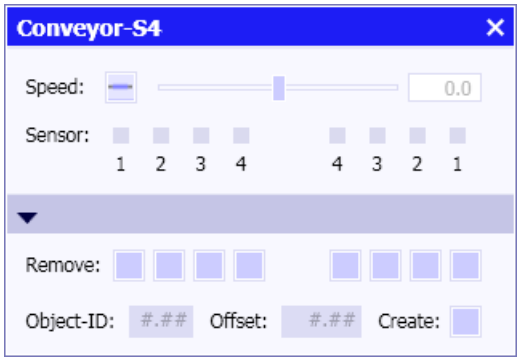
Name		Value
MaterialType		
MaterialList		
InitNbrOfObjects		0
Clearance	[mm]	0.0
TEMPLATE		Conveyor-S4
HIERARCHY		CONVEYOR

Operating window

In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider and monitor the status of the maximum eight sensors.

In the extended operating window you can remove objects that are currently detected by one of the eight sensors from the conveyor section (*Remove*). In addition, you can place new objects on the conveyor section using the *Create* pushbutton (*Create*). The object with the specified object ID (*Object-ID*) is then placed on the conveyor at the specified distance (*Offset*) from connector A. If no object ID is specified, the system searches the material lists (*MaterialList* additional parameter) for an object of the type specified by the *MaterialType* additional parameter.

This simulates manual interventions in the conveyor system such as the removal and placement of objects.



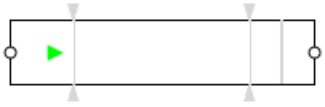
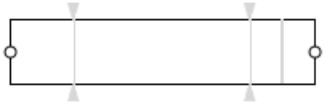
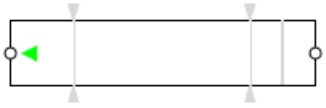
Conveyor-S4-Stopper – Straight conveyor with four sensors and stopper

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the objects. When the simulation is running the current transport direction is indicated by green arrows in the symbol.

Table 8-49 Indication of the current transport direction in the symbol

	Transport in the reference direction
	No transport
	Transport against the reference direction

The width of the symbol is scalable. The set width corresponds to the length of the conveyor.

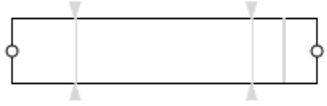
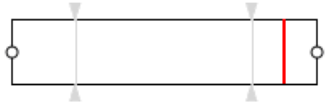
Function

The *Conveyor-S4-Stopper* component type simulates a straight conveyor section of a given length with a stopper at a given position on the conveyor section. The speed at which objects are moved over this conveyor is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter).

Between one and four sensors can be positioned on the conveyor relative to each of the two connectors *A* and *B*. When an object is within the detection range of a sensor, the corresponding hidden binary output *Sensor1* to *Sensor4* is set. The ID of the detected object can be obtained via the corresponding hidden integer outputs *SensorId1* to *SensorId4*.

The stopper can be activated and deactivated to stop objects at the stopper position in both conveyor directions. When a stopper is activated, its color in the symbol changes to red.

Table 8-50 Representation of the stopper in the symbol

	Deactivated stopper
	Activated stopper

Parameters

The behavior of the component can be set by means of parameters:

- *NominalSpeed*
  Nominal conveyor speed; adjustable online
- *StopperPosition*
  Position of the stopper relative to connector *A*
- *NbrOfSensorsA*
  Number of sensors relative to connector *A* (1 to 4)
- *SensorPositionA*
  Position of the corresponding sensor relative to connector *A* of the symbol.
- *NbrOfSensorsB*
  Number of sensors relative to connector *B* (1 to 4)
- *SensorPositionB*
  Position of the corresponding sensor relative to connector *B* of the symbol.

When the simulation is running, the defined sensors are shown in the correct position in the symbol. When a sensor is activated, its color changes to yellow.

The parameters, their units and default values are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0
StopperPosition	[mm]	0.0
NbrOfSensorsA		1
▾ SensorPositionA [1]		...
SensorPositionA1	[mm]	0.0
NbrOfSensorsB		1
▾ SensorPositionB [1]		...
SensorPositionB1	[mm]	0.0

Additional parameters

The *Conveyor-S4* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

Further additional parameters can be used to place objects on the conveyor for the simulation. These are placed on the conveyor after the start of the simulation, starting at end *A*:

- *MaterialType*
  The type of object initially placed on the conveyor
- *MaterialList*
  The name of the material list from which the object is to be taken. If this parameter is left blank, all the material lists in the simulation project are searched.

- *InitNbrOfObjects*
  The number of objects that are to be placed
- *Clearance*
  The distance between the objects that are to be placed:

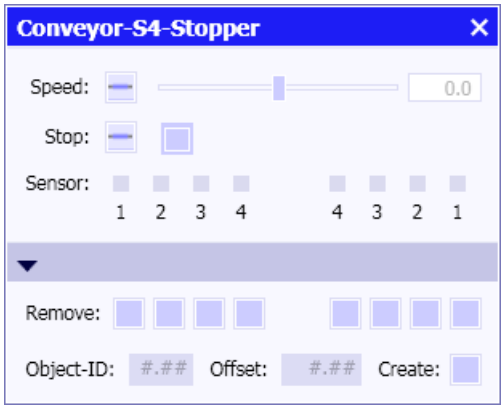
Name	Value
MaterialType	
MaterialList	
InitNbrOfObjects	0
Clearance	0.0
TEMPLATE	Conveyor-S4-Stopper
HIERARCHY	CONVEYOR

Operating window

In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider and monitor the status of the maximum eight sensors. The stopper can be activated and deactivated by means of a switch.

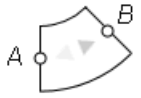
In the extended operating window you can remove objects that are currently detected by one of the eight sensors from the conveyor section (*Remove*). In addition, you can place new objects on the conveyor section using the *Create* pushbutton (*Create*). The object with the specified object ID (*Object-ID*) is then placed on the conveyor at the specified distance (*Offset*) from connector A. If no object ID is specified, the system searches the material lists (*MaterialList* additional parameter) for an object of the type specified by the *MaterialType* additional parameter.

This simulates manual interventions in the conveyor system such as the removal and placement of objects.



ConveyorCurve45-R60 – 45° curved conveyor (radius 60 pixels)

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the objects. When the simulation is running the current transport direction is indicated by green arrows in the symbol.

Function

The *ConveyorCurve45-R60* component type is used to simulate a curved conveyor. The conveyor section forms a 45° arc. The component symbol cannot be scaled. The radius of the arc is set to 60 pixels and cannot be changed. The effective radius of the component is determined by the scale set for the chart. For example, a scale of 1 pix: 100 mm gives an effective radius of 6 meters.

The speed at which objects are moved over this conveyor is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter).

Parameter

The nominal conveyor speed can be set by means of a parameter. Its unit and default value are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0

Additional parameters

The *ConveyorCurve45-R60* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

Name	Value
TEMPLATE	ConveyorCurve45-R60
HIERARCHY	CONVEYOR

Operating window

In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider.



ConveyorCurve45-R100 – 45° curved conveyor (radius 100 pixels)

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the objects. When the simulation is running the current transport direction is indicated by green arrows in the symbol.

Function

The *ConveyorCurve45-R100* component type is used to simulate a curved conveyor. The conveyor section forms a 45° arc. The component symbol cannot be scaled. The radius of the arc is set to 100 pixels and cannot be changed. The effective radius of the component is determined by the scale set for the chart. For example, a scale of 1 pix: 100 mm gives an effective radius of 10 meters.

The speed at which objects are moved over this conveyor is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter).

Parameters

The nominal conveyor speed can be set by means of a parameter. Its unit and default value are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0

Additional parameters

The *ConveyorCurve45-R100* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

Name	Value
TEMPLATE	ConveyorCurve45-R100
HIERARCHY	CONVEYOR

Operating window

In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider



ConveyorCurve45-R200 – 45° curved conveyor (radius 200 pixels)

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the objects. When the simulation is running the current transport direction is indicated by green arrows in the symbol.

Function

The *ConveyorCurve45-R200* component type is used to simulate a curved conveyor. The conveyor section forms a 45° arc. The component symbol cannot be scaled. The radius of the arc is set to 200 pixels and cannot be changed. The effective radius of the component is determined by the scale set for the chart. For example, a scale of 1 pix: 100 mm gives an effective radius of 20 meters.

The speed at which objects are moved over this conveyor is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter).

Parameters

The nominal conveyor speed can be set by means of a parameter. Its unit and default value are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0

Additional parameters

The *ConveyorCurve45-R200* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

Name	Value
TEMPLATE	ConveyorCurve45-R200
HIERARCHY	CONVEYOR

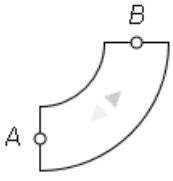
Operating window

In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider



ConveyorCurve90-R60 – 90° curved conveyor (radius 60 pixels)

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the objects. When the simulation is running the current transport direction is indicated by green arrows in the symbol.

Function

The *ConveyorCurve45-R60* component type is used to simulate a curved conveyor. The conveyor section forms a 90° arc. The component symbol cannot be scaled. The radius of the arc is set to 60 pixels and cannot be changed. The effective radius of the component is determined by the scale set for the chart. For example, a scale of 1 pix: 100 mm gives an effective radius of 6 meters.

The speed at which objects are moved over this conveyor is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter).

Parameters

The nominal conveyor speed can be set by means of a parameter. Its unit and default value are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0

Additional parameters

The *ConveyorCurve90-R60* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

Name	Value
TEMPLATE	ConveyorCurve90-R60
HIERARCHY	CONVEYOR

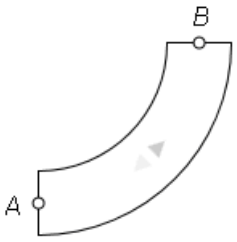
Operating window

In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider.



ConveyorCurve90-R100 – 90° curved conveyor (radius 100 pixels)

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the objects. When the simulation is running the current transport direction is indicated by green arrows in the symbol.

Function

The *ConveyorCurve90-R100* component type is used to simulate a curved conveyor. The conveyor section forms a 90° arc. The component symbol cannot be scaled. The radius of the arc is set to 100 pixels and cannot be changed. The effective radius of the component is determined by the scale set for the chart. For example, a scale of 1 pix: 100 mm gives an effective radius of 10 meters.

The speed at which objects are moved over this conveyor is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter).

Parameters

The nominal conveyor speed can be set by means of a parameter. Its unit and default value are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0

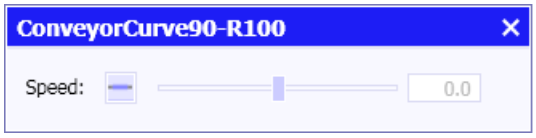
Additional parameters

The *ConveyorCurve90-R100* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

Name	Value
TEMPLATE	ConveyorCurve90-R100
HIERARCHY	CONVEYOR

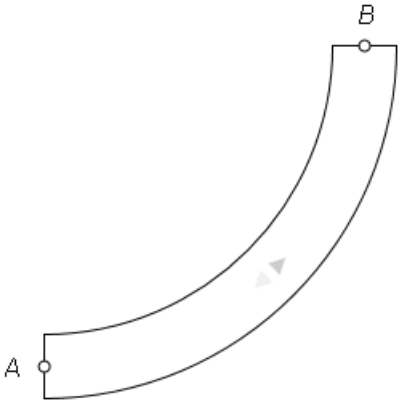
Operating window

In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider.



ConveyorCurve90-R200 – 90° curved conveyor (radius 200 pixels)

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the objects. When the simulation is running the current transport direction is indicated by green arrows in the symbol.

Function

The *ConveyorCurve90-R200* component type is used to simulate a curved conveyor. The conveyor section forms a 90° arc. The component symbol cannot be scaled. The radius of the arc is set to 200 pixels and cannot be changed. The effective radius of the component is determined by the scale set for the chart. For example, a scale of 1 pix: 100 mm gives an effective radius of 20 meters.

The speed at which objects are moved over this conveyor is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter).

Parameters

The nominal conveyor speed can be set by means of a parameter. Its unit and default value are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0

Additional parameters

The *ConveyorCurve90-R200* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

Name	Value
TEMPLATE	ConveyorCurve90-R200
HIERARCHY	CONVEYOR

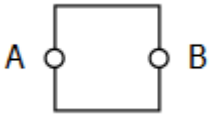
Operating window

In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider.



ConveyorAdapter – Connection component for cross-chart connection

Symbol



Function

The *ConveyorAdapter* component type is used to simulate a straight conveyor section. The speed at which vehicles are moved over this conveyor section is set at the hidden *Speed* input of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter).

This simple conveyor section can be used in particular to make cross-chart connections.

Parameter

The behavior of the component can be set by means of a parameter:

- *NominalSpeed*
  Nominal conveying speed in m/s; can be changed online.

Additional parameters

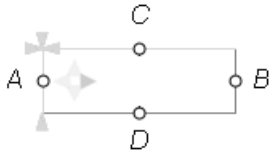
The *ConveyorAdapter* component includes the additional parameters *TEMPLATE* and *HIERARCHY*, which can be used to control generation of the drive level. You can find more information on this in section: Using templates (Page 882).

Operating window

In the operating window, you can specify the conveyor speed as a percentage value of the nominal conveyor speed (*NominalSpeed*) using a slider.

90DegreeTransfer – 90-degree transfer

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the objects on the conveyor section *A – B*. When the simulation is running the current transport direction is indicated by green arrows in the symbol. The direction of transfer (to *D* or *C*) is also indicated by green arrows.

The width of the symbol is scalable. The set width corresponds to the length of the 90-degree transfer. As with all component types in the *CONVEYOR* library, the height of the symbol is fixed at 40 pixels. The effective width of the 90-degree transfer is therefore determined by the scale set for the chart.

The location of positions C and D for the transfer of objects can be moved. To do so, hold down the ALT key and drag the connection point C or D with the mouse (with the left mouse button held down) to the desired position.

Function

The *90DegreeTransfer* component type is used to simulate a 90-degree transfer for moving objects from conveyor section *A – B* to a conveyor connected at connector *C* or *D* without rotating them.

The speed at which objects are moved over the conveyor section *A – B* is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter). Correspondingly, the speed at which the objects are transferred to

the transfer section *C – D* is set at the hidden analog input *TransferSpeed* as a percentage of the configurable nominal transfer speed (*NominalTransferSpeed* parameter). The reference direction of the transfer sections is set from connector C to the transport section or from the transport section to connector *D*.

Between one and four sensors can be positioned on the transfer relative to each of the four connectors *A*, *B, C* and *D*. When an object is within the detection range of a sensor, the corresponding hidden binary output *Sensor1* to *Sensor4* is set. The ID of the detected object can be obtained via the corresponding hidden integer outputs *SensorId1* to *SensorId4*.

Parameters

The behavior of the component can be set by means of parameters:

- *NominalSpeed*
  Nominal conveyor speed; adjustable online
- *NominalTransferSpeed*
  Nominal transfer speed; adjustable online
- *PositioningAccuracy*
  Accuracy with which the object must be positioned for transfer to the transfer section connecting to *C* or *D*
- *NbrOfSensorsA*
  Number of sensors relative to connector *A* (1 to 4)
- *SensorPositionA*
  Position of the corresponding sensor relative to connector *A* of the symbol.
- *NbrOfSensorsB*
  Number of sensors relative to connector *B* (1 to 4)
- *SensorPositionB*
  Position of the corresponding sensor relative to connector *B* of the symbol.
- *NbrOfSensorsC*
  Number of sensors relative to connector *C* (1 to 4)
- *SensorPositionC*
  Position of the corresponding sensor relative to connector *C* of the symbol.
- *NbrOfSensorsD*
  Number of sensors relative to connector *D* (1 to 4)
- *SensorPositionD*
  Position of the corresponding sensor relative to connector *D* of the symbol.

When the simulation is running, the defined sensors are shown in the correct position in the symbol. When a sensor is activated, its color changes to yellow.

The parameters, their units and default values are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0
PositioningAccuracy	[mm]	100.0
NbrOfSensorsA		1
SensorPositionA [1]		...
SensorPositionA1	[mm]	0.0
NbrOfSensorsB		1
SensorPositionB [1]		...
SensorPositionB1	[mm]	0.0
NbrOfSensorsC		1
SensorPositionC [1]		...
SensorPositionC1	[mm]	0.0
NbrOfSensorsD		1
SensorPositionD [1]		...
SensorPositionD1	[mm]	0.0

Additional parameters

The *90DegreeTransfer* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

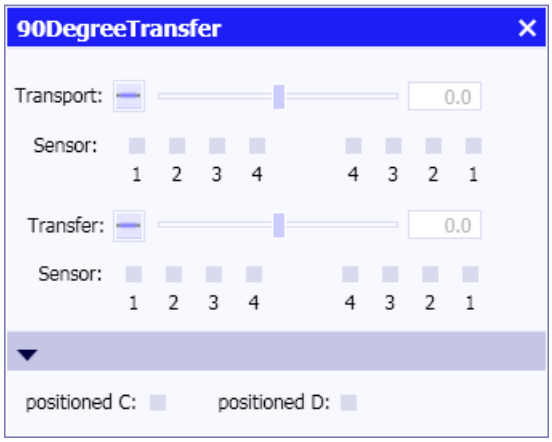
You also need to specify how many objects can be present on the conveyor section at any one time.

Name	Value
MaxObjects	8
TEMPLATE	90DegreeTransfer
HIERARCHY	CONVEYOR

Operating window

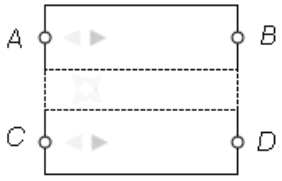
In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) and the transfer speed as a percentage of the nominal transfer speed (*NominalTransferSpeed*) using a slider and monitor the status of the maximum eight sensors.

The extended operating window shows whether the object is close enough to connector C or D to be transferred.



CrossOver – Crossover

Symbol



The two gray arrow heads in the symbol indicate the reference direction for movement of the objects over sections A – B and C – D. When the simulation is running the current transport direction is indicated by green arrows in the symbol.

The width of the symbol is scalable. The set width corresponds to the length of the crossover.

Function

The *CrossOver* component type simulates a crossover. The speed at which objects are moved over this switch is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter).

The switch is operated by means of the two binary inputs *Switch_CB_DA* and *Switch_AD_BC*. If the *Switch_CB_DA* input is set (True), the object is no longer transported over section *A – B* but over section *A – D* (transport in reference direction) or section *B – C* (transport against reference direction), depending on the transport direction. Correspondingly, if the *Switch_AD_BC* input is set (True), the object is no longer transported over section *C – D* but over section *C – B* (transport in reference direction) or over section *A – D* (transport against reference direction), depending on the transport direction. Setting both inputs at the same time deactivates the switch, which means that transport is stopped.

Parameters

The nominal speed at which the object is transported is set by means of the *NominalSpeed* parameter. Its unit and default value are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0

Additional parameters

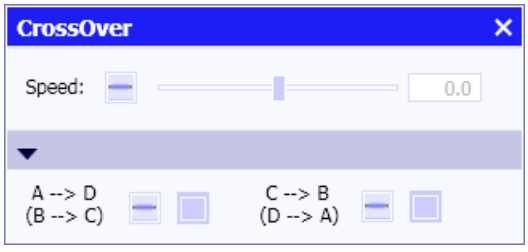
The *CrossOver* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

Name	Value
TEMPLATE	CrossOver
HIERARCHY	CONVEYOR

Operating window

In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider and monitor the status of the maximum eight sensors.

In the extended operating window you can set the two binary inputs *Switch_CB_DA* and *Switch_AD_BC* by means of a slider to set the transport direction of the crossover manually.



Lifter

A lifter is a conveyor section that can be moved vertically through several levels. In SIMIT a lifter is made up of the base component of the *LifterBase* type and up to eight expansion components of the *LifterExtension* type. The base component simulates the lifter in the base level, while each additional level is simulated by an expansion component. A lifter that can be moved through up to eight additional levels can be simulated in this way.

You do not have to position the base component and the associated expansion components on the same chart. You can create a separate chart for each level, for example, and distribute the lifter components over the individual charts.

The following points apply to the parameter assignment of the base component:

- There are no restrictions on the name of the base component.
- The *NbrOfExtensions* parameter indicates the number of additional levels and hence also the number of expansion components used.
- In the base component you have to specify the position (*LevelPosition*) for each level, which means the level above the base level in millimeters. The base level is by definition at level zero. All level values must be positive and increase with the level number.

The following points should be borne in mind when assigning the parameters for the expansion components:

- The name of the expansion component is formed from the name of the base component, the '#' character, and a number indicating the level.
- The *Level* parameter must correspond to the level number. The numbering of the levels starts with one.
- The *BaseName* parameter is the name of the base component.
- The length of the lifter is determined by the width of the base component.

Note

All expansion components must be exactly the same width as the base component. Otherwise, a message will be displayed pointing to this inconsistency when the simulation is started.

The drives and the lifter sensors are fully implemented in the base component. This means expansion components do not work without the base component.

LifterBase – Lifter (base component)

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the objects. When the simulation is running the current transport direction is indicated by green arrows in the symbol.

The width of the symbol is scalable. The set width corresponds to the length of the lifter.

Function

The *LifterBase* component type is used to simulate the base station of a lifter. The speed at which objects are moved over the conveyor section is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter). Similarly, the lifting speed, which means the speed at which the lifter is moved to the different levels, is set at the hidden analog input *LifterSpeed* as a percentage of the configurable nominal lifting speed (*NominalLifterSpeed* parameter).

To move objects in and out at the various lifter levels, the lifter must be moved to the appropriate level and stop flush with that level. Flushness must be set with a positioning accuracy Δ that is the same for all levels (*PositioningAccuracy* parameter). The lifter stops flush with a level *H* if the following applies for its level h:

$H - \Delta \leq h \leq H + \Delta$

Between one and four sensors can be positioned on the lifter relative to each of the two connectors *A* and *B*. When an object is within the detection range of a sensor, the corresponding hidden binary output *Sensor1* to *Sensor4* is set. The ID of the detected object can be obtained via the corresponding hidden integer outputs *SensorId1* to *SensorId4*.

Note

The base component has further hidden inputs for exchanging signals with extension components. The corresponding signals are assigned to these inputs by default. Do not change these default settings, otherwise the lifter components will not behave in the intended way in the simulation.

Parameters

The behavior of the component is configurable.

- *NominalSpeed*
  Nominal conveyor speed; adjustable online
- *NominalLifterSpeed*
  Nominal lifting speed; adjustable online
- *NbrOfExtensions*
  Number of additional levels (1 to 8)
- *LevelPosition*
  Level of the corresponding additional level
- *PositioningAccuracy*
  Positioning accuracy within which the level is deemed to have been reached.
- *NbrOfSensorsA*
  Number of sensors relative to connector A (1 to 4)
- *SensorPositionA*
  Position of the corresponding sensor relative to connector *A* of the symbol.

- *NbrOfSensorsB*
  Number of sensors relative to connector B (1 to 4)
- *SensorPositionB*
  Position of the corresponding sensor relative to connector *B* of the symbol.

When the simulation is running, the defined sensors are shown in the correct position in the symbol. When a sensor is activated, its color changes to yellow.

The parameters, their units and default values are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0
NominalLifterSpeed	[m/s]	1.5
NbrOfExtensions		1
▼ LevelPosition [1]		...
LevelPosition1	[mm]	0.0
PositioningAccuracy	[mm]	100.0
NbrOfSensorsA		1
▼ SensorPositionA [1]		...
SensorPositionA1	[mm]	0.0
NbrOfSensorsB		1
▼ SensorPositionB [1]		...
SensorPositionB1	[mm]	0.0

Additional parameters

The *LifterBase* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

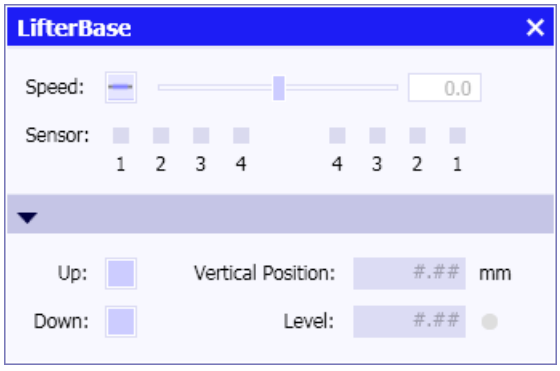
You also need to specify how many objects can be present on this conveyor section at any one time.

Name	Value
MaxObjects	8
TEMPLATE	LifterBase
HIERARCHY	CONVEYOR

Operating window

In the operating window, you can use a slider to set the transport speed as a percentage of the nominal conveyor speed (*NominalSpeed*) and you can monitor the status of the sensors.

In the extended operating window you can move the lifter up or down by means of pushbuttons. The current lifter position and level are displayed.



LifterExtension – Lifter (extension component)

Symbol



The gray arrow head in the symbol indicates the reference direction for movement of the objects. The width of the symbol is scalable. The set width corresponds to the length of the lifter rail.

Note

The width of the extension component must be the same as the width of the base component.

Function

The *LifterExtension* component type is used to simulate a lifter at one of the accessible levels. A component of this type can only be used in combination with a component of the *LifterBase* type.

Parameters

The behavior of the component is configurable:

- *Level*
  Level at which the component is located. The numbering starts at one.
- *BaseName*
  Name of the corresponding base component

The parameters, their units and default values are shown in the figure below.

Name	Value
BaseName	
Level	1

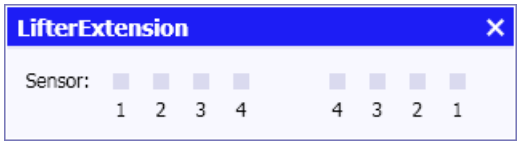
Additional parameters

Specify how many objects can be present on the section defined by the component at any one time. This parameter should be set to the same value as for the base component.

Name	Value
MaxObjects	8

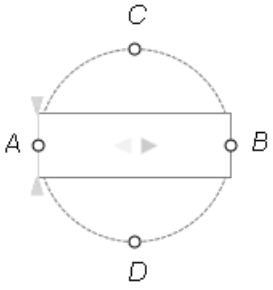
Operating window

In the operating window, you can monitor the status of the maximum eight sensors.



Turntable-R60 – Turntable

Symbol



The gray arrow head in the symbol indicates the reference direction for transport of the object on the turntable. When the simulation is running the current transport direction is indicated by green arrows in the symbol.

Function

The *Turntable-R60* component type simulates a turntable. A turntable rotates a conveyor section through multiples of 90°. Using a turntable, objects can be transported onwards in the original conveyor direction or at right angles to that direction.

The speed at which objects are moved on the turntable is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed*

parameter). Correspondingly, the rotation speed is set at the hidden analog input *RotationSpeed* as a percentage of the configurable nominal rotation speed (*NominalRotationSpeed* parameter).

The component symbol cannot be scaled. The radius of the turntable is thus fixed at 60 pixels and cannot be changed. Correspondingly, the length of the turntable is fixed at 120 pixels. The effective turntable length of the component is determined by the scale set for the chart. For example, a scale of 1 pix: 100 mm gives an effective length of 12 meters.

Between one and four sensors can be positioned on the turntable relative to each of the two connectors *A* and *B*. When an object is within the detection range of a sensor, the corresponding hidden binary output *Sensor1* to *Sensor4* is set. The ID of the detected object can be obtained via the corresponding hidden integer outputs *SensorId1* to *SensorId4*.

Parameters

The behavior of the component can be set by means of parameters:

- *NominalSpeed*
  Nominal conveyor speed; adjustable online
- *NominalRotationSpeed*
  Nominal rotation speed; adjustable online
- *PositioningAccuracy*
  Accuracy with which the turntable must be positioned for the object to be transferred
- *NbrOfSensorsA*
  Number of sensors relative to connector *A* (1 to 4)
- *SensorPositionA*
  Position of the corresponding sensor relative to connector *A* of the symbol.
- *NbrOfSensorsB*
  Number of sensors relative to connector *B* (1 to 4)
- *SensorPositionB*
  Position of the corresponding sensor relative to connector *B* of the symbol.

The parameters, their units and default values are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0
NominalRotationSpeed	[°/s]	90.0
PositioningAccuracy	[°]	1.0
NbrOfSensorsA		1
▼ SensorPositionA [1]		...
SensorPositionA1	[mm]	0.0
NbrOfSensorsB		1
▼ SensorPositionB [1]		...
SensorPositionB1	[mm]	0.0

Additional parameters

The *TurnTable-R60* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

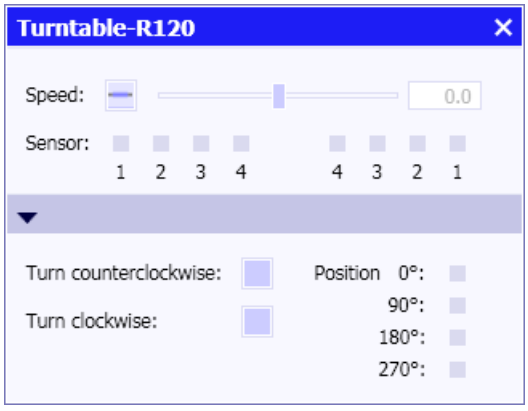
Specify how many objects can be located on the turntable at the same time.

Name	Value
MaxObjects	8
TEMPLATE	Turntable-R60
HIERARCHY	CONVEYOR

Operating window

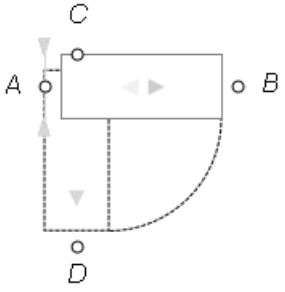
In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider and monitor the status of the maximum 8 sensors.

In the extended operating window you can rotate the turntable manually by 90° in both directions by means of pushbuttons. The end position in each case is displayed.



Swiveltable-R120 – Swivel table

Symbol



The gray arrow head in the symbol indicates the reference direction for transport of the object on the swivel table. When the simulation is running the current transport direction is indicated by green arrows in the symbol.

Function

The *Swiveltable-R120* component type simulates a swivel table that can be used to rotate a conveyor section through 90°. Using a swivel table, objects can be transported onwards in the original conveyor direction or at right angles to that direction.

The speed at which objects are moved on the swivel table is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter). Correspondingly, the rotation speed is set at the hidden analog input *RotationSpeed* as a percentage of the configurable nominal rotation speed (*NominalRotationSpeed* parameter).

The component symbol cannot be scaled. This means that the radius of the swivel table, i.e. the length of the swivel table, is fixed at 120 pixels and cannot be changed. The effective swivel table length of the component is determined by the scale set for the chart. For example, a scale of 1 pixel: 100 mm gives an effective length of 12 meters.

Between one and four sensors can be positioned on the swivel table relative to each of the two connectors *A* and *B*. When an object is within the detection range of a sensor, the corresponding hidden binary output *Sensor1* to *Sensor4* is set. The ID of the detected object can be obtained via the corresponding hidden integer outputs *SensorId1* to *SensorId4*.

Note

This component must not be mirrored.

Parameters

The behavior of the component can be set by means of parameters:

- *NominalSpeed*
  Nominal conveyor speed; adjustable online
- *NominalRotationSpeed*
  Nominal rotation speed; adjustable online
- *PositioningAccuracy*
  Accuracy with which the turntable must be positioned for the object to be transferred
- *NbrOfSensorsA*
  Number of sensors relative to connector *A* (1 to 4)
- *SensorPositionA*
  Position of the corresponding sensor relative to connector *A* of the symbol.
- *NbrOfSensorsB*
  Number of sensors relative to connector *B* (1 to 4)
- *SensorPositionB*
  Position of the corresponding sensor relative to connector *B* of the symbol.

The parameters, their units and default values are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0
NominalRotationSpeed	[°/s]	90.0
PositioningAccuracy	[°]	1.0
NbrOfSensorsA		1
SensorPositionA [1]		...
SensorPositionA1	[mm]	0.0
NbrOfSensorsB		1
SensorPositionB [1]		...
SensorPositionB1	[mm]	0.0

Additional parameters

The Swiveltable-R120 component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

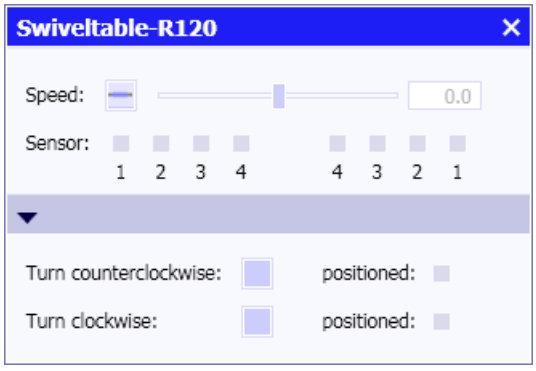
Specify how many objects can be located on the swivel table at the same time.

Name	Value
MaxObjects	8
TEMPLATE	Swiveltable-R120
HIERARCHY	CONVEYOR

Operating window

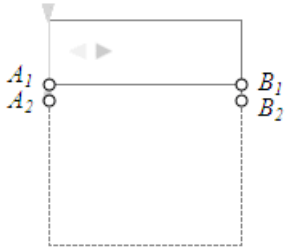
In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider and monitor the status of the maximum 8 sensors.

In the extended operating window you can rotate the swivel table manually through 90° by means of a pushbutton. Another pushbutton allows you to rotate it back to its original position. Indicators show when it has reached the end position.



TransferCarriage – Transfer carriage

Symbol



The height and width of this component are scalable. The width sets the length of the transfer carriage, the height sets the length of the transfer path.

The gray arrow head in the symbol indicates the reference direction for transport of the object on the swivel table. When the simulation is running the current transport direction is indicated by green arrows in the symbol.

Function

The *TransferCarriage* component type is used for simulating a transfer carriage. Objects can be moved onto the transfer carriage from both sides. It can then be moved to another position at right angles to the transport direction and the objects can then be transported onwards via connectors A_i and B_i.

The number of connectors A_i, B_i present on the left and right of the component is configurable. A maximum of 16 connectors on each side is possible; by default there are two connectors on each side. A_1, A_2, B_1, B_2. Each connector marks a position that can be approached by the transfer carriage at right angles to the transport direction.

The speed at which objects are transported on the transfer carriage is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter). Correspondingly, the transfer speed is set at the hidden analog input *TransferSpeed* as a percentage of the configurable nominal transfer speed (*NominalTransferSpeed* parameter).

Between one and four sensors can be positioned on the transfer carriage relative to each of the two connector sides *A* and *B*. When an object is within the detection range of a sensor, the corresponding hidden binary output *Sensor1* to *Sensor4* is set. The ID of the detected object can be obtained via the corresponding hidden integer outputs *SensorId1* to *SensorId4*.

Parameters

The behavior of the component can be set by means of parameters:

- *NominalSpeed*
  Nominal conveyor speed; adjustable online
- *NominalTransferSpeed*
  Nominal transfer speed; adjustable online

- *PositioningAccuracy*
  Accuracy with which the transfer carriage must be positioned for the object to be transferred.
- *NbrOfConnectorsA*
  Number of connectors on the left side (A)
- *NbrOfConnectorsB*
  Number of connectors on the right side (B)
- *NbrOfSensorsA*
  Number of sensors relative to connector side *A* (1 to 4)
- *SensorPositionA*
  Position of the corresponding sensor relative to connector side *A* of the symbol.
- *NbrOfSensorsB*
  Number of sensors relative to connector side *B* (1 to 4)
- *SensorPositionB*
  Position of the corresponding sensor relative to connector side *B* of the symbol.

The parameters, their units and default values are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0
NominalTransferSpeed	[m/s]	1.5
PositioningAccuracy	[mm]	50.0
NbrOfConnectorsA		2
NbrOfConnectorsB		2
NbrOfSensorsA		1
▾ SensorPositionA [1]		...
SensorPositionA1	[mm]	0.0
NbrOfSensorsB		1
▾ SensorPositionB [1]		...
SensorPositionB1	[mm]	0.0

Additional parameters

The *TransferCarriage* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

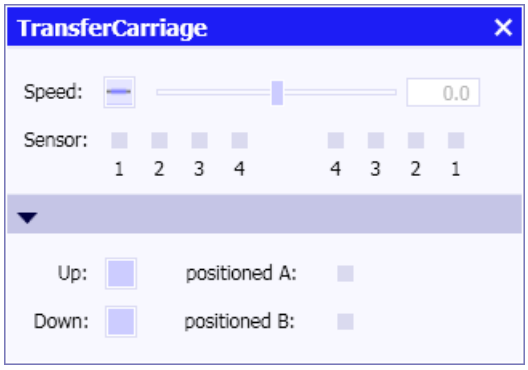
Specify how many objects can be located on the transfer carriage at the same time.

Name	Value
MaxObjects	8
TEMPLATE	TransferCarriage
HIERARCHY	CONVEYOR

Operating window

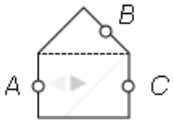
In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider and monitor the status of the maximum 8 sensors.

In the extended operating window you can move the transfer carriage manually in both directions to the next position by means of pushbuttons (*Up*, *Down*). The end position in each case is displayed. Indicators show when the transfer carriage has reached a connector position on one or the other side.



SpurConveyor-1 – Diagonal feed with one spur

Symbol



The gray arrow head in the symbol indicates the reference direction for transport of the object on the diagonal feed. When the simulation is running the current transport direction is indicated by green arrows in the symbol.

Function

The *SpurConveyor-1* component type simulates a diagonal feed with one spur. Objects can be transported along section *A – C* or along the spur section *A – B*.

The speed at which objects are moved on the diagonal feed is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter). To switch conveying to the spur section, the hidden binary input *Switch* is set to one (True).

Parameter

The nominal conveyor speed can be set by means of a parameter. Its unit and default value are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0

Additional parameters

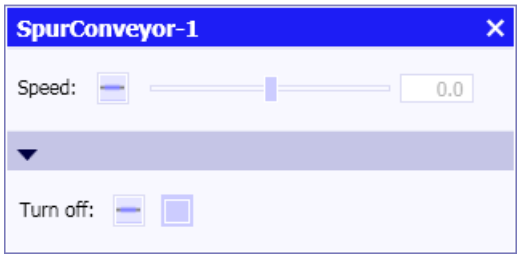
The *SpurConveyor-1* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in the section: Using templates (Page 882).

Name	Value
TEMPLATE	SpurConveyor-1
HIERARCHY	CONVEYOR

Operating window

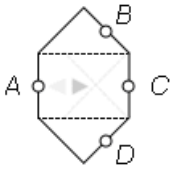
In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider.

In the extended operating window you can change to the spur section by means of a switch.



SpurConveyor-2 – Diagonal feed with two spurs

Symbol



The gray arrow head in the symbol indicates the reference direction for transport of the object on the diagonal feed. When the simulation is running the current transport direction is indicated by green arrows in the symbol.

Function

The *SpurConveyor-2* component type simulates a diagonal feed with two alternative spurs. Objects can be transported along section *A – C* or along one of the two spur sections *A – B* or *A – D* .

The speed at which objects are moved on the diagonal feed is set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter). To switch conveying to one of the two spur sections, one of the two

hidden binary inputs *Switch_AB or Switch_AD* is set to one (True): *Switch_AB* switches to spur section *A – B* and *Switch_AD* switches to spur section *A – D*. If one of the two inputs is set, setting the other one has no effect. There will be no switching to either spur section if both inputs are set at the same time.

Parameter

The nominal conveyor speed can be set by means of a parameter. Its unit and default value are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0

Additional parameters

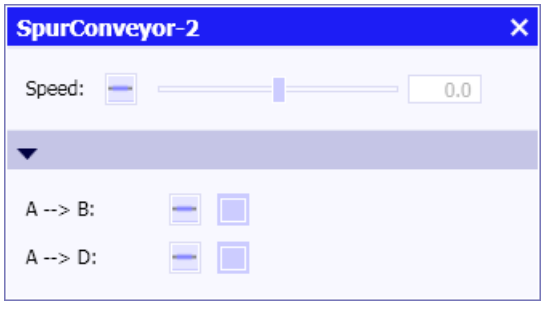
The *SpurConveyor-2* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in the section: Using templates (Page 882).

Name	Value
TEMPLATE	SpurConveyor-2
HIERARCHY	CONVEYOR

Operating window

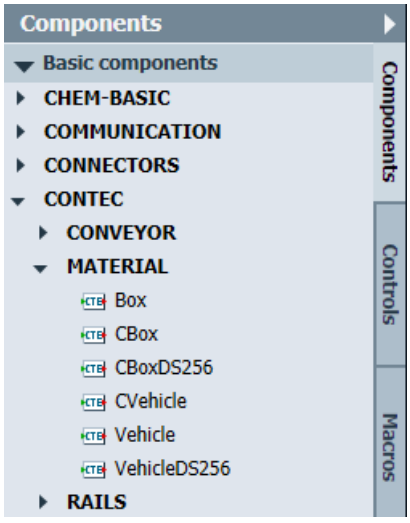
In the operating window, you can set the conveyor speed as a percentage of the nominal conveyor speed (*NominalSpeed*) using a slider.

In the extended operating window you can change to the corresponding spur section *A – B* or *A – D* by means of a switch.



8.3.3.4 Component types for simulating objects

The *MATERIAL* directory of the *CONTEC* library contains component types for simulating objects. Vehicles (*Vehicle*) can be used with component types in the *RAILS* directory to simulate vehicular conveyor systems. Boxes (*Box*) can be used to simulate objects for component types in the CONVEYOR directory.



You create material lists with these components. You can find more information on the topic in the section: Scalability of objects (Page 879).

The size of a component (width and height of the symbol) is defined in the material list in millimeters.

Box – Simple object

Symbol



Function

The *Box* component type is used to simulate an object with a rectangular outline, such as a pallet or box. The component type defines the dimensions of the object. In the resolution 1:20 the standard dimensions correspond to the dimensions of a Euro pallet (1200 × 800 mm).

CBox – Colored object

Symbol



Function

The *CBox* component type is used to simulate an object with a rectangular outline, such as a pallet or box. The component type defines the dimensions of the object. In the resolution 1:20 the standard dimensions correspond to the dimensions of a Euro pallet (1200 × 800 mm).

When the simulation is running the symbol of a component of this type can be displayed in one of 8 different colors. The color is determined with the three binary inputs *R*, *G* and *B* of the component by mixing together the 3 primary colors red, green and blue.

Table 8-51 Color chart for CBox

Input R	Input G	Input B	Display
False	False	False	
False	False	True	
False	True	False	
False	True	True	
True	False	False	
True	False	True	
True	True	False	
True	True	True	

CBoxDS256 – Colored object with data storage

Symbol



Function

The *CBoxDS256* component type is used to simulate an object with a rectangular outline, such as a pallet or box. The component type defines the dimensions of the object. In the resolution 1:20 the standard dimensions correspond to the dimensions of a Euro pallet (1200 × 800 mm).

When the simulation is running the symbol of a component of this type can be displayed in one of 8 different colors. The color is determined with the three binary inputs *R*, *G* and *B* of the component by mixing together the 3 primary colors red, green and blue.

Table 8-52 Color chart for *CBoxDS256*

Input R	Input G	Input B	Display
False	False	False	
False	False	True	
False	True	False	
False	True	True	
True	False	False	
True	False	True	
True	True	False	
True	True	True	

In addition, a data storage with a configurable size (*SizeOfStorage* parameter) is defined for components of this type. This memory represents the mobile data storage for an object. It is created as a state vector *MDS* with the data type *byte* and can be up to 256 bytes in size. Component types for writing to and reading this memory are located in the *SENSORS* library in the *RFID directory*.

Parameters

The size of the data storage can be set by means of the *SizeOfStorage* parameter. The default setting is one.

Name	Value
SizeOfStorage	1

Vehicle

Symbol



Function

The *Vehicle* component type is used for simulating vehicles, such as those for electric overhead monorail systems. The vehicle is assumed to have a rectangular outline. The component type defines the dimensions of the outline. In the resolution 1:20 the standard dimensions correspond to the dimensions of a Euro pallet (1000 × 300 mm).

The speed at which these vehicles move in the rail network in the simulation is generally determined by the individual rail sections. However, the speed can also be set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter). The two speed settings for a component are added together.

A component of this type can also be configured so that the vehicle starts off again with a delay if it has previously been stopped by a holdup. This function is only effective if the speed is set by the conveyor section.

Parameters

The behavior of the component can be set by means of parameters:

- *NominalSpeed*
  Nominal speed of the vehicle; adjustable online
- *StartUpDelay*
  Start-up delay
- *SensorRange*
  Detection range of the sensors

The parameters, their units and default values are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0
StartUpDelay	[s]	0.0
SensorRange	[%]	100.0

Additional parameters

The *Vehicle* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

Name	Value
TEMPLATE	Vehicle
HIERARCHY	VEHICLES

CVehicle

Symbol



Function

The *CVehicle* component type is used for simulating vehicles, such as those for electric overhead monorail systems. The vehicle is assumed to have a rectangular outline. The component type defines the dimensions of the outline. In the resolution 1:20 the standard dimensions correspond to the dimensions of a Euro pallet (1000 × 300 mm).

The speed at which these vehicles move in the rail network in the simulation is generally determined by the individual rail sections. However, the speed can also be set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter). The two speed settings for a component are added together.

A component of this type can also be configured so that the vehicle starts off again with a delay if it has previously been stopped by a holdup. This function is only effective if the speed is set by the conveyor section.

When the simulation is running the symbol of a component of this type can be displayed in one of 8 different colors. The color is determined with the 3 binary inputs *R*, *G* and *B* of the component by mixing together the 3 primary colors red, green and blue.

Table 8-53 Color chart for *CVehicle*

Input R	Input G	Input B	Display
False	False	False	
False	False	True	
False	True	False	
False	True	True	
True	False	False	
True	False	True	
True	True	False	
True	True	True	

Parameters

The behavior of the component can be set by means of parameters:

- *NominalSpeed*
  Nominal speed; adjustable online
- *StartUpDelay*
  Start-up delay
- *SensorRange*
  Detection range for sensors

The parameters, their units and default values are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0
StartUpDelay	[s]	0.0
SensorRange	[%]	100.0

Additional parameters

The *CVehicle* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

Name	Value
TEMPLATE	CVehicle
HIERARCHY	VEHICLES

VehicleDS256 – Vehicle with data storage

Symbol



Function

The *VehicleDS256* component type is used for simulating vehicles, such as those for electric overhead monorail systems. The vehicle is assumed to have a rectangular outline. The component type defines the dimensions of the outline. In the resolution 1:20 the standard dimensions correspond to the dimensions of a Euro pallet (1000 × 300 mm).

The speed at which these vehicles move in the rail network in the simulation is generally determined by the individual rail sections. However, the speed can also be set at the hidden analog input *Speed* of the component as a percentage of the configurable nominal conveyor speed (*NominalSpeed* parameter). The two speed settings for a component are added together.

A component of this type can also be configured so that the vehicle starts off again with a delay if it has previously been stopped by a holdup. This function is only effective if the speed is set by the conveyor section.

In addition, a data storage with a configurable size (*SizeOfStorage* parameter) is defined for components of this type. This memory represents the mobile data storage for a vehicle. It is created as a state vector *MDS* with the data type *byte* and can be up to 256 bytes in size. Component types for writing to and reading this memory are located in the *SENSORS* library in the *RFID* directory.

Parameters

The behavior of the component can be set by means of parameters:

- *NominalSpeed*
  Nominal speed; adjustable online
- *StartUpDelay*
  Start-up delay
- *SensorRange*
  Detection range for sensors
- *SizeOfStorage*
  Size of the mobile data storage

The parameters, their units and default values are shown in the figure below.

Name		Value
NominalSpeed	[m/s]	2.0
StartUpDelay	[s]	0.0
SensorRange	[%]	100.0
SizeOfStorage		1

Additional parameters

The *VehicleDS256* component includes the additional parameters *TEMPLATE and HIERARCHY*, which can be used to control generation of the drive level. You can find additional information on this in section: Using templates (Page 882).

Name	Value
TEMPLATE	VehicleDS256
HIERARCHY	VEHICLES

8.3.4 Creating custom component types for material handling simulation

The SIMIT component type editor *CTE* allows you to create your own component types, which use the mechanisms of the conveyor technology simulation solution procedure. You can thus expand the functions of the components supplied with the *CONTEC* library, for example, by implementing more complex transport processes that are not covered by the supplied library components and thus enhance your conveyor technology simulations. You can also create your own component types from scratch to extend the *CONTEC* library and adapt it to your specific conditions.

Three aspects must be considered when creating component types:

- The topological aspects,
- The connection to the solution procedure, and
- The specific behavior of the component.

The topological aspect is covered by appropriate extensions to the component type definition. Special connection types are provided for the connection to the solution procedure.

In addition, the general descriptions of component properties in the "SIMIT – Component Type Editor" manual form the basis for creating component types for material handling simulation. The general properties also apply in full to these component types.

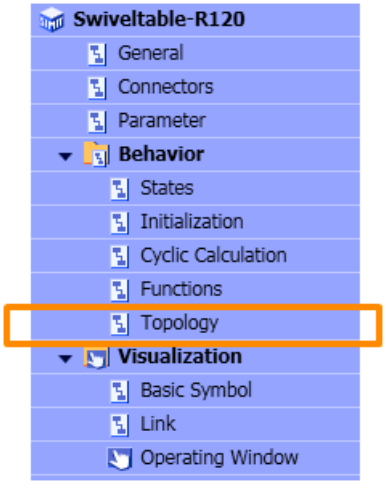
8.3.4.1 Topological properties

The topology of the modeled conveyor system is automatically determined when compiling a simulation project from interconnected material handling components. Each component must therefore provide relevant topological information about itself, which means information about how it is to be treated topologically in the system. From a topological point of view it is necessary to know how the reference directions for variables in a material handling system are defined for a component and how data exchange between the component and the solution procedure is set up.

Elements of the material handling system with topological information are:

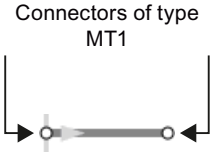
- Segments of conveyor sections
- Branches
- Boundaries

Relationships with the topological connectors of the material handling component must be defined for each of these elements. To do this, open the topology editor by double-clicking on the *Topology* entry in the component type navigation menu.



Topological connectors (connection type MT1)

The *MT1* connection type indicates the connectors that are used to connect material handling components to one another. The topology of the conveyor system is derived from interconnected connectors of this type and the topological information about the individual handling equipment components. *MT1* is purely a topological connector type, which means it does not carry any signals. Connectors of this type are simply referred to below as "topological connectors". Topological connectors are represented by circles.



They can be joined by superimposing the connectors. Once linked, both connectors are hidden.



If topological connectors are connected with each other by a connecting line in the chart editor, then the connected connectors are hidden. They are still visible only as pale shadows. Connectors of the MT1 type cannot be joined more than once; they can only ever establish a 1:1 connection.



Topology of a segment

The topology description of a segment is used to set its reference direction. For example, the definition

```
FROM a TO b;
```

defines a segment with a reference direction from the starting point `a` to the end point `b`. The starting point and end point can be defined in the component type as a connector of type *MT1* or as a branch or boundary.

Speeds are positive in the reference direction. The reference direction is also the preferred direction, which is used by the solution procedure to determine the calculation order.

Topology of a branch

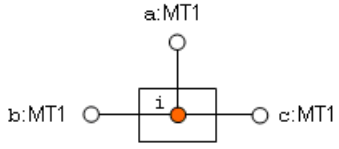
A branch as an internal node in a component type is defined as follows in the topology description:

```
INTERNAL_NODE i;
```

The topological connectors of the component must also be assigned to this branch. Three connectors can be assigned in the topological description as per the example below:

```
FROM i TO a;
FROM i TO b;
FROM i TO c;
```

The three connectors a, b and c must be defined as type *FLN1* connectors in the component type. The figure below shows a diagram of the resulting topological structure of the component type:



A branch can also be defined as multiple segments having a common starting or end point that is defined as an *MT1* topological connector.

Topology of a boundary

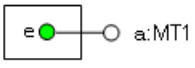
In the topological description of a component type a boundary point as an external node `e` is defined as follows:

```
EXTERNAL_NODE e;
```

This external node must be also be linked to at least one topological connector of the component. The topology description is, for example, to be completed as

```
FROM e TO a;
```

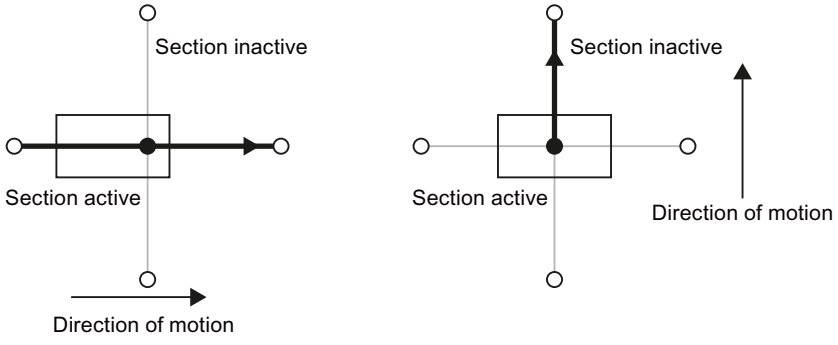
. The topological connector a must be defined in the component type as a connector of type *MT1*. The figure below shows a diagram of the resulting topological structure of the component type.



Determining the next section

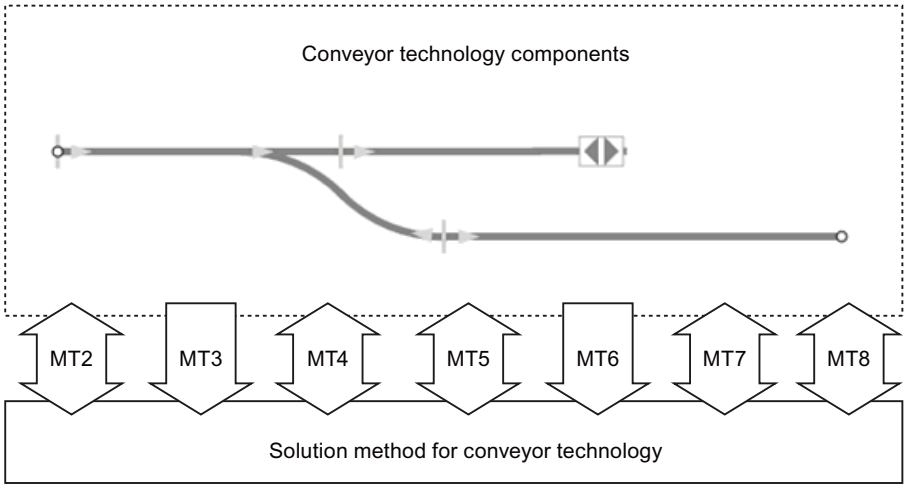
An object is only ever assigned to one segment in the conveyor system network. Transport can only take place if the transition to the next segment is unambiguously defined. This means only one possible subsequent segment can be identified as active at branching points; all other alternative segments must be inactive.

An exception occurs at a branch as an internal node of a component, if the outline of an object covers the branch and only one of the segments connected to the branch is active. In this case the object is assigned to this active segment.



8.3.4.2 Connection to the solution procedure

The segments, branches and boundary points of a component and the solution procedure for the material handling exchange data via special connectors. There are seven different connection types *MT2* to *MT8* available for this purpose. Their use in material handling components is explained in the sections below.



Connectors of connection types *MT2* to *MT8* establish a connection to the solution procedure and must therefore be set as **hidden** on the component symbol. In the connector properties, the usage "*Only in property view*" or "*Only in CTE*" must be set.

Property	Value
Usage	Property view only
Visibility Default	

Connectors of connection type MT2 to MT7 should be defined in the OUT direction, a connector of connection type MT8 in the IN direction.

See also

Connection type MT2 for segments (Page 949)

Connection type MT3 for stoppers (Page 950)

Connection type MT4 for sensors (Page 950)

Connector type MT5 for placing objects (Page 951)

Connection type MT6 for positions (Page 952)

Connector type MT7 for parameter assignment of objects (Page 953)

Connector type MT8 for transferring objects at boundary points (Page 954)

Connection type MT2 for segments

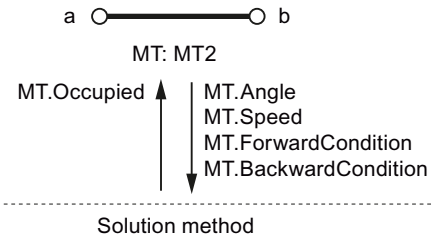
A segment can exchange variables with the solution procedure via a connector of type *MT2*. For a connector with the name *MT* the topological description of the segment is completed as follows:

```
FROM a TO b: MT;
```

The connector must always be defined in the OUT direction. It connects the component to the solution procedure to exchange various variables that are relevant to the segment between the solution procedure and the component.

Name	Connection type	Direction	Dimension
MT	MT2	OUT	1

The figure below shows the input and output signals with the direction of data exchange between the component and the solution procedure.



The output signals for the segment are set or calculated in the component and sent to the solution procedure:

1. **Angle** (integer)
   The angle in degrees of the segment of a circle that describes this segment. Positive angles rotate in a clockwise direction, negative angles rotate in a counterclockwise direction. For a straight connection the angle should be set to zero.
   This setting is only evaluated when the simulation is initialized; changing the value during the cyclical calculation has no effect.
2. **Speed** (analog)
   The speed in m/s at which objects are to be moved over this segment. Positive values move it in the preferred direction, negative values move it against the preferred direction.
3. **ForwardCondition** (integer)
   Specifies whether this segment is active in the preferred direction (0) or not (–1).
4. **BackwardCondition** (integer)
   Specifies whether this segment is active against the preferred direction (0) or not (–1).

The component receives information about the conveyor section from the solution procedure via the input signal:

1. **Occupied** (bool)
   Specifies whether the outline of one or more objects is in contact with this segment.

Connection type *MT2* is used in the *Rail-S4* and *Conveyor-S4* library components, for example.

Connection type MT3 for stoppers

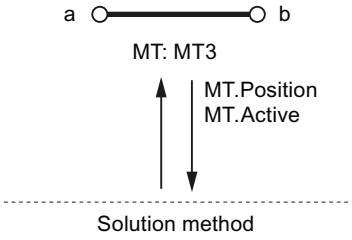
A segment can exchange variables with the solution procedure via a connector of type *MT3*. For a connector with the name *MT* the topological description of the segment is completed as follows:

```
FROM a TO b: MT;
```

The connector must always be defined in the OUT direction. It connects the component to the solution procedure to exchange various variables between the solution procedure and the component.

Name	Connection type	Direction	Dimension
MT	MT3	OUT	1

The figure below shows the input and output signals with the direction of data exchange between the component and the solution procedure.



The output signals for the segment are set or calculated in the component and sent to the solution procedure:

1. **Position** (analog)
   The position of the stopper on the segment as a percentage of the total length of the segment.
   This setting is only evaluated when the simulation is initialized; changing the value during the cyclical calculation has no effect.
2. **Active** (binary)
   Indicates whether the stopper is active (True) or inactive (False).

Connector type *MT3* is used in the *Conveyor-S4-Stopper* library component, for example.

Connection type MT4 for sensors

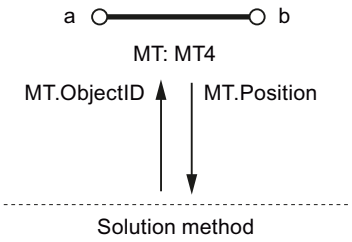
A segment can exchange variables with the solution procedure via a connector of type *MT4*. For a connector with the name *MT* the topological description of the segment is completed as follows:

```
FROM a TO b: MT;
```

The connector must always be defined in the OUT direction. It connects the component to the solution procedure to exchange various variables between the solution procedure and the component.

Name	Connection type	Direction	Dimension
MT	MT4	OUT	1

The figure below shows the input and output signals with the direction of data exchange between the component and the solution procedure.



The output signals for the segment are set or calculated in the component and sent to the solution procedure:

1. **Position** (analog)
   The position of the sensor on the segment as a percentage of the total length of the segment. This setting is only evaluated when the simulation is initialized; changing the value during the cyclical calculation has no effect.

The component receives information about the segment from the solution procedure via the input signal:

1. **ObjectID** (integer)
   The ID of the object that is currently detected by the sensor. If no object is detected, this value is zero.

Connection type *MT4* is used in the *Rail-S4* and *Conveyor-S4* library components, for example.

Connector type MT5 for placing objects

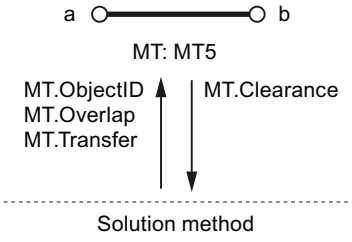
A connector of connection type *MT5* can be used for placing objects on a segment. For a connector with the name *MT*, the topological definition of the segment should be completed as follows:

```
FROM a TO b: MT;
```

The connector must always be defined in the OUT direction. It connects the component to the solution procedure to exchange various variables between the solution procedure and the component.

Name	Connection type	Direction	Dimension
MT	MT5	OUT	1

The figure below shows the input and output signals with the direction of data exchange between the component and the solution procedure.



The output signals for the segment are set or calculated in the component and sent to the solution procedure:

1. **ObjectID** (integer)
   The ID of the object to be placed on this segment. The component generally obtains this ID by means of appropriate function calls. You can find information on this in the following sections:
   - _MT.GetObjectByName (Page 957)
   - _MT.GetObjectByType (Page 957)
2. **Overlap** (analog)
   Overlap in millimeters in the preferred direction. This can be used to move the placement position in relative terms. The placement position must always be on the segment, however.
3. **Transfer** (bool)
   When this signal is set to *True*, the solution procedure starts the placement process. The component can only set the signal for exactly one cycle, after which it must return to *False*.

The component receives information about the segment from the solution procedure via the input signals:

1. **Clearance** (analog)
   The available length of segment in millimeters viewed in the preferred direction.

Connection type *MT5* is used in the *Rail-S4* and *Conveyor-S4* library components, for example.

Connection type MT6 for positions

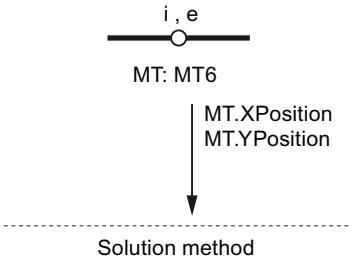
The positions of branches as internal nodes and boundary points as external nodes have to be configured individually. The topological definition of a node must be supplemented with a connector via which the position is sent to the solution procedure, for a connector *MT* as follows:

```
EXTERNAL_NODE  E: MT;
INTERNAL_NODE I: MT;
```

The connector must always be defined in the OUT direction. It connects the component to the solution procedure to exchange various variables that are relevant to the node between the solution procedure and the component.

Name	Connection type	Direction	Dimension
MT	MT6	OUT	1

The figure below shows the input and output signals with the direction of data exchange between the component and the solution procedure.



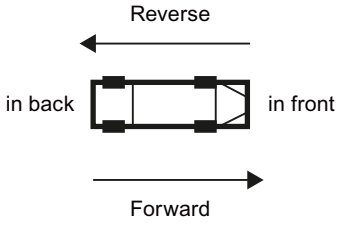
The output signals for the node are set or calculated in the component and sent to the solution procedure:

1. **XPosition** (analog)
   The X position of the node in millimeters relative to the top left corner of the component. This setting is only evaluated when the simulation is initialized; changing the value during the cyclical calculation has no effect.
2. **YPosition** (analog)
   The Y position of the node in millimeters relative to the top left corner of the component. This setting is only evaluated when the simulation is initialized; changing the value during the cyclical calculation has no effect.

Connector type MT7 for parameter assignment of objects

An object can exchange variables with the solution procedure via a connector of type *MT7*. No topological description is necessary.

The orientation of the object is dependent on its configuration and corresponds to the way the basic symbol is displayed in the CTE.

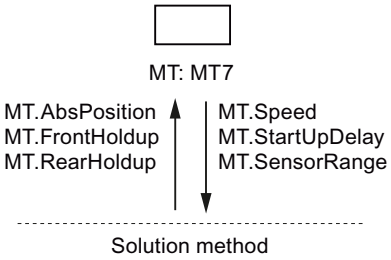


The connector of type *MT7* must always be defined in the OUT direction. It connects the component to the solution procedure to exchange various variables that are relevant to the object between the solution procedure and the component.

A component can have no more than one MT7 type connector.

Name	Connection type	Direction	Dimension
MT	MT7	OUT	1

The figure below shows the input and output signals with the direction of data exchange between the component and the solution procedure.



The output signals for the object are set or calculated in the component and sent to the solution procedure:

1. **Speed** (analog)
   The speed of the object established by its drive in m/s. A positive speed moves the vehicle forwards, a negative speed moves it backwards.
   This speed is replaced by a speed set by the rail if applicable.
2. **StartUpDelay** (analog)
   The start-up delay in seconds. If the object is held up by the solution procedure, the object waits for this time before moving again once the hold-up is cleared.
3. **SensorRange** (analog)
   The percentage of the object length that is detected by a sensor. This value must be between 0 and 100%.

The component receives information about the object from the solution procedure via the input signals:

1. **AbsPosition** (analog)
   Reserved for subsequent development.
2. **FrontHoldup** (binary)
   If this signal is set to *True*, there is a stopper or another object directly in front of the object.
3. **RearHoldup** (binary)
   If this signal is set to *True*, there is a stopper or another object directly behind the object.

Connector type *MT7* is used in the *Vehicle* (Page 940) library component, for example.

Connector type MT8 for transferring objects at boundary points

Objects can be transferred from the solution procedure to a component and vice versa at boundary points as external nodes. Once a component rather than the solution procedure has control over the object, the position of the object cannot be changed by the solution procedure. The object remains visible on the chart, however, and can now be positioned by the component by means of corresponding system functions. You can find additional information on this in section: _MT.SetPosition (Page 959).

This allows more complex transport operations, which cannot be simulated with a path-based solution procedure, to be implemented in conveyor components.

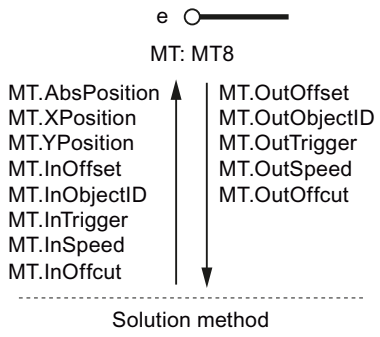
A boundary point as an external node can exchange variables with the solution procedure via a connector of type *MT8*. For a connector with the name *MT* the topological description of the external node is completed as follows:

```
EXTERNAL_NODE  E: MT;
```

The connector must always be defined in the IN direction. It connects the component to the solution procedure to exchange various variables between the solution procedure and the component.

Name	Connection type	Direction	Dimension
MT	MT8	IN	1

The figure below shows the input and output signals with the direction of data exchange between the component and the solution procedure.



The output signals for the external node are set or calculated in the component and sent to the solution procedure:

1. **OutOffset** (analog)
   The length of the available section, viewed from the connector into the component. A negative value means that viewed from the component, an object is projecting onto the adjacent section controlled by the solution procedure; in this case the value corresponds to the length of the projecting part of the object. All values are given in millimeters.
2. **OutObjectID** (integer)
   The object ID of the object that viewed from the component is projecting onto the adjacent section controlled by the solution procedure. The value is zero if no object is projecting.
3. **OutTrigger** (binary)
   The component sets this signal to *True* for exactly one cycle to show that the adjacent section controlled by the solution procedure has to take over the object.
4. **Out.Speed** (analog)
   The speed in m/s at which the object is transferred.
5. **OutOffcut** (analog)
   The time that the current cycle could not include because the object had reached the transfer node, measured in seconds. The solution procedure has to take this time into account in the next cycle to calculate an additional movement.

The component receives information from the solution procedure via the input signals:

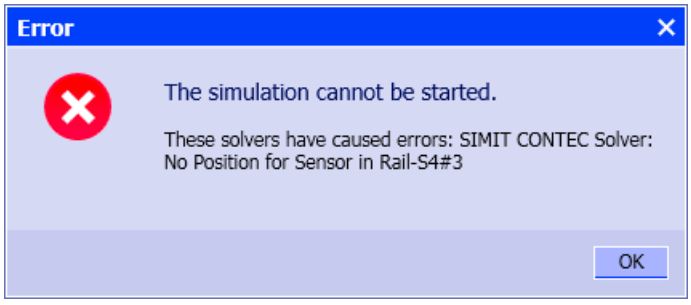
1. **AbsPosition** (analog)
   Reserved for subsequent developments.
2. **XPosition** (analog)
   The relative X position of the connector linked to the node relative to the top left corner of the component in millimeters.
3. **YPosition** (binary)
   The relative Y position of the connector linked to the node relative to the top left corner of the component in millimeters.
4. **InOffset** (analog)
   The length of the available section, viewed from the component connector, in the direction of the adjacent section outside the component. A negative value means that viewed from the adjacent section controlled by the solution procedure, an object is projecting into this component; in this case the value corresponds to the length of the projecting part of the object. All values are given in millimeters.

5. **InObjectID** (integer)
   The object ID of the object that viewed from the section controlled by the solution procedure is projecting into the component. The value is zero if no object is projecting.
6. **InTrigger** (binary)
   The solution procedure sets this signal to *True* for exactly one cycle to show that the component has to take over the object.
7. **InSpeed** (analog)
   The speed in m/s at which the object was last moved by the solution procedure.
8. **InOffcut** (analog)
   The time that the current cycle could not include because the object had reached the transfer node, measured in seconds. The component has to take this time into account in the next cycle in order to calculate an additional movement.

Connector type *MT8* is used in the *TransferCarriage* library component, for example.

Errors in the component code

If interfaces to the solution procedure are created in material handling components but not all necessary signals are made available, the simulation start request is denied and a corresponding error message displayed.



8.3.4.3 System functions

If you create your own component types you can utilize various system functions to access objects in the simulation.

All functions described below return an error code as the return value. In order for a function to be able to supply return values in a variable that is passed, this variable must be defined as a field (vector), in this case with the dimension one. Refer to the table below for the meaning of this return value.

Table 8-54 Return values of conveyor technology system functions

Value	Meaning
0	No error
–2	The object cannot be positioned directly because it is currently under solution procedure control.
–3	The object cannot be positioned or rotated because it is not currently assigned to a conveyor section.

Value	Meaning
–10	There is no object with the specified ID.
–11	The object with the specified ID is not available.[1)]
–12	There is no object with the specified name.
–13	The object with the specified name is not available.
–14	There is no object of the specified type.
–15	An object of the specified type is not available.
–20	The component has no signal with the specified name.
–21	The specified signal type cannot be used in this context.
–22	The address range of the byte array to be read or written has been exceeded.
–1000	General error

[1] An object is not available if it is reserved for being assigned to a conveyor section or if it is assigned to a conveyor section.

_MT.GetObjectByName

The *_MT.GetObjectByName* function returns the ID of the object with the instance name *name* if it is available as an object, otherwise it returns the value –1. The return value type is *long*.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- name (string)
  The instance name of the object.

_MT.GetObjectByType

The *_MT.GetObjectByType* function returns the ID of an object of component type *type* if it is available as an object, otherwise it returns the value –1. The return value type is *long*.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- type (string)
  The type of object required.
- store (string)
  The name of the material list in which the object is located. If an empty string is passed here, all existing lists are searched.

_MT.Restore

The *_MT.Restore* function releases an object. The return value is *void*.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be released.
- delay (bool)
  Indicates whether the return should be delayed by one cycle so that the object is available for immediate reuse.

_MT.GetWidth

The *_MT.GetWidth* function returns the width of an object in millimeters. The return value type is *double*.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be queried.

_MT.GetHeight

The *_MT.GetHeight* function returns the height of an object in millimeters. The return value type is *double*.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be queried.

_MT.GetDepth

The *_MT.GetDepth* function returns the depth of an object in millimeters. The return value type is *double*.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be queried.

_MT.GetAngle

The *_MT.GetAngle* function returns the angle in degrees of an object relative to the tangent of the segment on which it is located (–180° .. +180°). The return value type is *integer*.

Note that this function can only be called for objects that are assigned to a conveyor section.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be queried.

_MT.SetPosition

The *_MT.SetPosition* function sets the position of an object relative to the position of the conveyor section component to which this object is assigned. The position is specified in millimeters relative to the top left corner of the component (in its starting position). The return value type is *void*.

Note that this function can only be called for objects that are assigned to a conveyor section but are not under solution procedure control.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be changed.
- x (analog)
  X position in millimeters relative to the conveyor section component.
- y (analog)
  Y position in millimeters relative to the conveyor section component.

_MT.SetAngle

The *_MT.SetAngle* function sets the angle in degrees of an object relative to the tangent of the segment on which it is located (–180° .. +180°). The return value type is *void*.

Note that this function can only be called for objects that are assigned to a conveyor section.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be changed.
- angle (integer)
  Angle in degrees relative to the tangent of the segment.

_MT.AddAngle

The *_MT.AddAngle* function increases the angle in degrees of an object relative to the tangent of the segment on which it is located (–180° .. +180°). The return value type is *void*.

Note that this function can only be called for objects that are assigned to a conveyor section.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be changed.
- angle (integer)
  Angle in degrees relative to the tangent of the segment.

_MT.SetHoldup

The *_MT.SetHoldup* function transfers to an object information about whether it is positioned directly in front of or behind a stopper or another object. The return value type is *void*.

Note that this function can only be called for objects that are assigned to a conveyor section but are not under solution procedure control.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be changed.
- front (binary)
  Information as to whether the object is blocked in front.
- rear (binary)
  Information as to whether the object is blocked to the rear.

_MT.GetBinaryValue

The *_MT.GetBinaryValue* function is used to query a binary variable (state, input, output) of an object. The return value type is *bool*.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be queried.
- property (string)
  The name of the variable to be queried

_MT.GetAnalogValue

The *_MT.GetAnalogValue* function is used to query an analog variable (discrete state, input, output) of an object. The return value type is *double*.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be queried.
- property (string)
  The name of the variable to be queried.

_MT.GetIntegerValue

The *_MT.GetIntegerValue* function is used to query an integer variable (state, input, output) of an object. The return value type is *long*.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be queried.
- property (string)
  The name of the variable to be queried.

_MT.GetByteValue

The *_MT.GetByteValue* function is used to query a byte (state) of an object. The return value type is *byte*.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be queried.
- property (string)
  The name of the state to be queried.

_MT.GetByteArray

The *_MT.GetByteArray* function is used to query a byte array (state) of an object. The return value type is *void*.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be queried.
- property (string)
  The name of the state vector to be read.
- stateoffset (long)
  The offset in bytes from which the state vector is to be read.
- buffer (byte[])
  Byte array in which the read byte is to be entered. If you specify a state vector for the component, enter it in the notation for new state values, which means prefixed with "@".
- bufferoffset (long)
  The offset in bytes from which the read bytes are to be entered.
- count (long)
  The number of bytes to be read.

Note

All arrays must be large enough to execute this operation; otherwise, unpredictable errors could occur in the simulation process.

_MT.SetBinaryValue

The *_MT.SetBinaryValue* function is used to write a binary variable (state, input) of an object. The return value type is *void*.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be described.
- property (string)
  The name of the state to be written.
- value (bool)
  The value to be written.

_MT.SetAnalogValue

The *_MT.SetAnalogValue* function is used to write an analog variable (discrete state, input) of an object. The return value type is *void*.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be described.
- property (string)
  The name of the variable to be written.
- value (double)
  The value to be written.

_MT.SetIntegerValue

The *_MT.SetIntegerValue* function is used to write an integer variable (state, input) of an object. The return value type is *void*.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be described.
- property (string)
  The name of the variable to be written.
- value (long)
  The value to be written.

_MT.SetByteValue

The *_MT.SetByteValue* function is used to write a byte (state) of an object. The return value type is *void*.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be described.
- property (string)
  The name of the state to be written.
- value (byte)
  The value to be written.

_MT.SetByteArray

The *_MT.SetByteArray* function is used to write a byte array (state) of an object. The return value type is *void*.

Specify values for the following parameters:

- error (long[])
  Error code
  You can find information about the error code in the table in the section: System functions (Page 956).
- id (long)
  The ID of the object to be described.
- property (string)
  The name of the state vector of the object to be described.
- stateoffset (long)
  The offset in bytes from which the state vector is to be written.
- buffer (byte[])
  Byte array containing the byte to be written.
- bufferoffset (long)
  The offset in bytes from which the byte array is to be transferred.
- count (long)
  The number of bytes to be written.

Note

All arrays must be large enough to execute this operation; otherwise, unpredictable errors could occur in the simulation process.

8.3.4.4 System variables

The system variables listed in the table below are available for modeling conveyor technology components. These variables are available in the component behavior description with read access only.

Table 8-55 System variables

Variable name	Meaning
_WIDTH	Width of the component in pixels
_HEIGHT	Height of the component in pixels
_SCALEX	Horizontal scaling of the component. The factor one is the default setting.
_SCALEY	Vertical scaling of the component. The factor one is the default setting.
_TECHSCALE	Scale of the chart on which the component is located. The number of millimeters corresponding to one pixel is specified.

Licensing

9

9.1 License model

The SIMIT license model is based on simulation tags. Simulation tags are calculated based on the total number of inputs and outputs as well as the state variables of all components used in the project. Because an implicit signal interconnection between two couplings takes on the function of a diagram in which these signals are connected by means of a component with at least one input and one output, such implicit interconnections in couplings are counted as two simulation tags.

Components that are used as pure data memories and do not contain any behavioral description are excluded from the counting of simulation tags when used on diagrams or in lists.
The following table shows the available SIMIT versions:

SIMIT version	Number of simulation tags
XS[1]	Up to 250
S	Up to 4,000
M	Up to 16,500
L	Up to 200,000
XL	Up to 1,000,000

[1] Only in conjunction with SIMIT unit hardware

Additional licenses

Additional licenses are required for the following:

- An upgrade to a new version of SIMIT
- Additional libraries (CHEM-BASIC, FLOWNET or CONTEC)

- Component Type Editor (CTE)
- SIMIT Virtual Controller (VC)
  - SIMIT VC: Controllers of the S7-300, S7-400 and S7-410 series are emulated with this license.
  - SIMIT VC-300: Only projects in which a S7-300 controller is configured are emulated with this license.
  - SIMIT VC-410E: Only projects in which a S7-410E or S7-410SIS controller is configured are emulated with this license.

Note

If no licenses for the S7-300 or S7-410E series are available, a search is carried out for the higher-order SIMIT VC license.

If no suitable licenses are found, the simulation starts in DEMO mode (Page 19).

The content of the system status lists supplied by the virtual controller does not correspond in every detail to that of the real controller types.

Behavior in SIMIT

Select the size variant when SIMIT starts up. You can also change the size variant later via Portal view > Start > Settings > Size variant.

The status bar shows the percentage of permitted simulation tags included in your simulation project.

Number of simulation tags exhausted up to	Consequence
80 %	One-time notification
110 % and more[1]	Simulation tags can be added: No Simulation project can be saved: No Simulation can be started: No

[1] SIMIT version XL: No limit. The PC performance capability determines the maximum size of a simulation project.

See also

Setting up WIBU license server (Page 968)

Help > License container (Page 981)

9.2 Setting up WIBU license server

Introduction

SIMIT directly accesses the dongle when it is inserted into the local computer.

If the dongle is inserted in a computer in the network, you must set up the access to the dongle. To do so, use the CodeMeter program on the server and the local computer.

Requirement

- The dongle is not inserted in the local computer but in a computer in the network.

Procedure

To access the CodeMeter control center directly on your computer, follow these steps:

1. Click on the following link: Server access (http://localhost:22352/$help/CmUserHelp/us/index.html?server_access.htm).

Procedure

Alternative procedure when CodeMeter is on a different computer.

To open the help, follow these steps:

1. Go to "Start > CodeMeter > CodeMeter Control Center".
   The "CodeMeter Control Center" opens.
2. Click on "WebAdmin".
3. Click on the "?" icon.
   The "CodeMeter User Help" opens.
4. Execute the steps listed under "Server access".

Result

SIMIT accesses a license on a dongle that is inserted in a computer in the network.

9.3 Setting up access permission to the license server

Introduction

If a license server has been set up, the access permissions need to be specified to grant the desired clients access to the available licenses.

Procedure

To access the CodeMeter control center directly on your computer, follow these steps:

1. Click on the following link: License access permissions (http://localhost:22352/$help/CmUserHelp/us/index.html?license_access_permissions.htm).

Alternative procedure

Alternative procedure when CodeMeter is on a different computer.

To open the help, follow these steps:

1. Go to "Start > CodeMeter > CodeMeter Control Center".
   The "CodeMeter Control Center" opens.
2. Click "WebAdmin".
3. Click the "?" icon.
   The "CodeMeter User Help" opens.
4. Execute the steps listed under "License access permissions".

9.4 Setting up a server search list on the client

Introduction

For the clients, it is possible to obtain licenses from different license servers. The order of the server search and the server call can be defined.

Procedure

To access the CodeMeter control center directly on your computer, follow these steps:

1. Click on the following link: Server search list (http://localhost:22352/$help/CmUserHelp/us/index.html?server_search_list.htm).

Alternative procedure

Alternative procedure when CodeMeter is on a different computer.

To open the help, follow these steps:

1. Go to "Start > CodeMeter > CodeMeter Control Center".
   The "CodeMeter Control Center" opens.
2. Click "WebAdmin".
3. Click the "?" icon.
   The "CodeMeter User Help" opens.
4. Execute the steps listed under "Server search list".

Menus and dialog boxes 10

10.1 Menus

10.1.1 Portal view > Start

The "Start" button provides access to the following functions:

- **Open existing project**
  Here, you can open an existing project. You can find additional information on this in the section: Project > Open... (Page 975)
- **Create new project**
  Here, you can create a new project. You can find additional information on this in the section: Project > New project ... (Page 974)
- **Retrieve project**
  Here, you can retrieve a project. You can find additional information on this in the section: Project > Retrieve (Page 977)
- **Retrieve sample project**
  You retrieve one of the supplied sample projects here. Select the desired sample project from the drop-down list. You can freely select the storage location of the retrieved sample project.
- **Close project**
  Closes the current project.
- **Getting started**
  This explains the first steps with SIMIT.
- **Installed software**
  Here, you are shown the currently installed SIMIT software. You can find additional information on this in the section: "Info" dialog box (Page 998)

- **Help**
  Here you access the SIMIT online help.
- **Settings**
  - **Size variant**
    Here you change the size variant for which you have purchased a license. The change takes effect when SIMIT is restarted.
  - **User interface language**
    Here you change the user interface language of SIMIT. The change takes effect when SIMIT is restarted.
  - **Show portal view on startup**
    Specifies whether SIMIT starts with the portal view or the project view.
  - **Query when exiting a simulation**
    Enables a prompt asking whether you really want to exit the simulation.
  - **Show note on value changes when exiting a simulation**
    Enables a notification that values were changed during the simulation.
  - **Query when loading a snapshot**
    Enables a prompt asking whether you really want to load the snapshot.

10.1.2 Portal view > Couplings

The "Couplings" button provides access to the following functions:

- **Add new coupling**
  Select a communication path by which data should be exchanged in SIMIT.
- **Assigning coupling signals**
  If the signals of the connectors can be uniquely assigned to a coupling, they can be updated with this function. This is required for name changes, for example.
- **SU administration**
  If "SIMIT Unit" is selected as the coupling, enter a name and the IP address of the SIMIT unit here.
- **Help**
  This is where you access the SIMIT online help.

See also

Options > Assign coupling signals (Page 981)

10.1.3 Portal view > Simulation model

The "Simulation model" button provides access to the following functions:

- **Add new chart**
  Add a new chart to your project. SIMIT switches to the project view in the chart editor.
- **Create new macro**
  Create a new macro. SIMIT switches to the project view in the Macro Editor.
- **Create new component type**
  Create a new component type. SIMIT switches to the project view in the Component Type Editor.
- **Help**
  Here you access the SIMIT online help.

10.1.4 Portal view > Automatic model creation

The "Automatic modelling" button provides access to the following functions:

- **Instantiate templates**
  Generate instances of templates from import files (*.xls, *.xlsx, *.txt, *.iea, *.xml) or from the simulation model. You can find additional information on this in section: Instantiation of templates from files or the simulation model (Page 286).
- **Create new template**
  Create a new template. SIMIT switches to the template editor for the project view. You can find additional information on this in section: Templates (Page 280).
- **Automated import**
  Imports models from the following sources:
  - XML (Page 300)
  - ZIP (Page 304)

  You can find additional information in the section Automated import (Page 299).
- **Bulk engineering import**
  Transfer the parameters and input defaults from a table to existing charts. You can find additional information on this in section: Bulk engineering (Page 305).
- **Help**
  This is where you access the SIMIT online help.

See also

Table import (Page 287)

IEA import (Page 291)

CMT import (Page 293)

10.1.5 Portal view > Diagnostics & visualization

The "Diagnostics & visualization" button provides access to the following functions:

- **Consistency check**
  Check your project for formal errors. The result of the consistency check is displayed in the project view. You can find additional information on this in section: Consistency check (Page 328).
- **Find & replace**
  Execute the "Find & replace" function in the current project. You can find additional information on this in section: Find & replace (Page 323).
- **Add new trend**
  Create a new trend. SIMIT switches to the project view in the Trend Editor. You can find additional information on this in section: Trends (Page 314).
- **Edit archive**
  Create an archive. A trend can be created from an archive. SIMIT switches to the project view in the Archive Editor. You can find additional information on this in section: Archive (Page 312).
- **Help**
  Here you access the SIMIT online help.

10.1.6 Project > New project ...

You can create a new project with this menu command. Enter the required data for this and make the settings in the input fields.

- **Project name**
  Enter a name for your project or accept the default name here.
- **Target folder**
  The storage location of the new project is preset here. To select a different storage location, click the "..." button.
- **Author**
  You can change the name of the author here. This entry is optional.
- **Comment**
  You can enter a comment of your choice about the project here. This entry is optional.
- **Engineering server**
  Select an engineering server that is connected to SIMIT.
- **Engineering project**
  Select an engineering project that will be imported from the engineering server.

Note

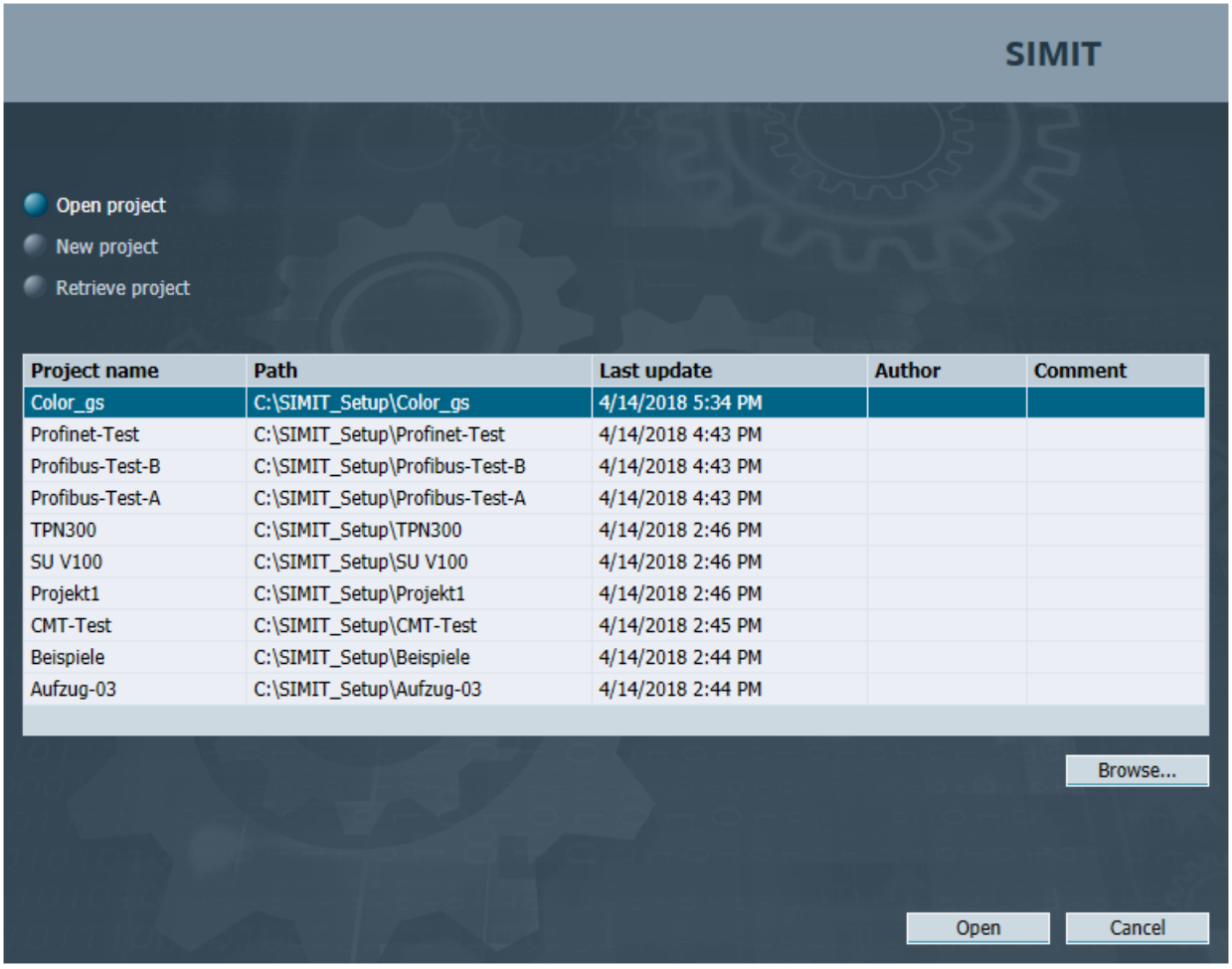
"Engineering server" and "Engineering project" can be subsequently selected or changed in the "Project manager".

You will find more information in section Properties (Page 248).

Click "Create" to create the new project and close the dialog box.

10.1.7 Project > Open...

Use this menu command to open the following dialog box:



The most recent projects and the sample project are listed here. The project name, location and date of the last update of the project are also displayed for each project. This information cannot be changed here.

If the desired project is not displayed here, you can search for it by clicking "Browse ...".

Open the project by selecting it and clicking the "Open" button.

10.1.8 Project > Close

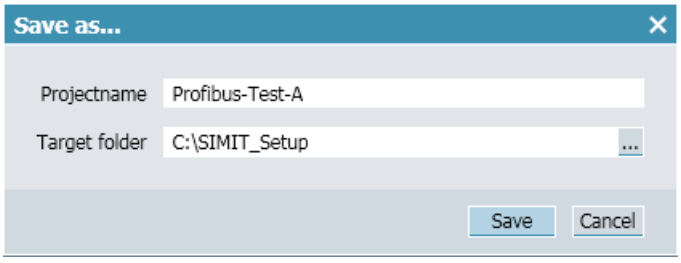
Use this menu command to close the current project.

10.1.9 Project > Save all

Use this menu command to save the current project.

10.1.10 Project > Save as...

Use this menu command to open the following dialog box:



Project name and target folder are preset.

- **Project name**
  Accept the preset name or enter another name for your project.
- **Target folder**
  Accept the preset storage location for your project or select another one by clicking "...".

Click "Save" to save the project and close the dialog box.

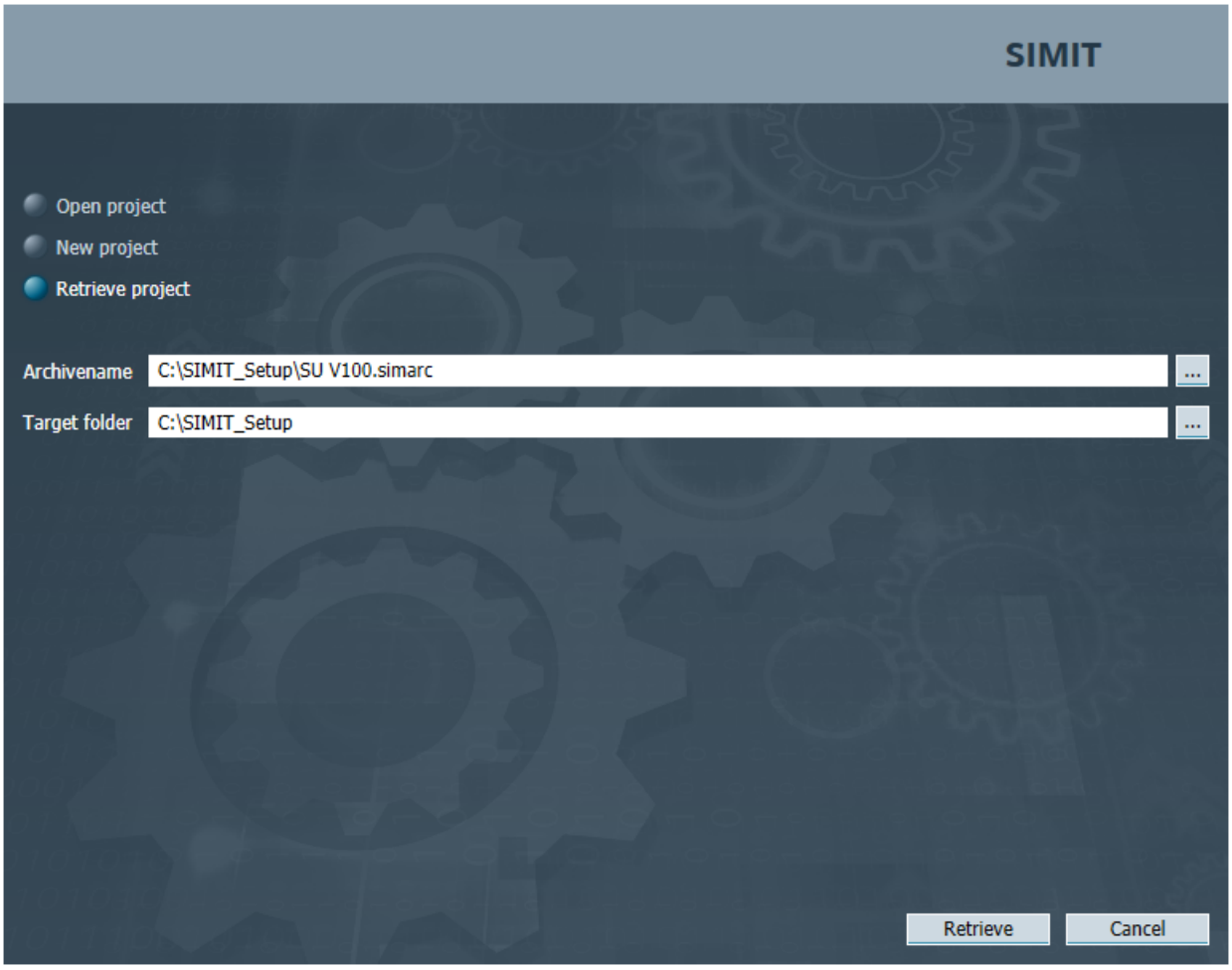
10.1.11 Project > Archive

This menu command opens the "Save as" dialog box, where you can enter a name and storage location for the current project. Then click the "Save" button to perform the archiving of the project.

The archived project receives the file extension "*.simarc".

10.1.12 Project > Retrieve

Use this menu command to open the following dialog box:



Here, you can retrieve an archived project.

- **Archive name**
  Click "..." to select an archived SIMIT project. Archived SIMIT projects have the file extension ".simarc".
- **Target folder**
  Click "..." to select the folder in which the project is to be stored after retrieval.

Click "Retrieve" to retrieve the selected project.

Note

An archived SIMIT project can contain executable code. Only retrieve projects from a trustworthy source.

10.1.13 Project > Analysis

Use this menu command to open a dialog box, in which you save a statistical evaluation of the data of a SIMIT project.

The file is saved in the "*.xslx" format.

10.1.14 Project > Exit

Use this menu command to close the current project.

If changes have not been saved in the project, a dialog box opens in which you are asked whether you want to save before exiting.

10.1.15 Edit > Cut

Use this menu command to remove the selected object.

10.1.16 Edit > Copy

Use this menu command to copy the selected object to the clipboard.

10.1.17 Edit > Paste

Use this menu command to paste the copied object from the clipboard to the current editor.

10.1.18 Simulation > Initialize

Use this menu command to start the consistency check. The result of the consistency check is displayed.

Note

Initialization does not start simulation.

10.1.19 Simulation > Start

Use this menu command to start the simulation.

You can find additional information on this in section: Actions during ongoing simulation (Page 34).

Before the simulation starts, you are asked whether you want to save the changes to the project and the consistency check is performed.

10.1.20 Simulation > Pause

Use this menu command to pause the simulation.

10.1.21 Simulation > Single Step

This menu command executes a single step in the simulation.

You can find additional information on this in the section: Executing a single step (Page 342).

10.1.22 Simulation > Exit

This menu command ends the simulation in progress and takes you to the project view.

10.1.23 Simulation > Snapshot

Use this menu command to make a snapshot of the simulation.

You can find additional information on this in the section: Actions during ongoing simulation (Page 34).

10.1.24 Simulation > Backtracks

Use this menu command to open the "Available backtracks" dialog box.

"Available backtracks" dialog box (Page 983)

10.1.25 Window > Tile Horizontally

Use this menu command to split the selected editor window horizontally.

10.1.26 Window > Tile Vertically

Use this menu command to split the selected editor window vertically.

10.1.27 Window > Unsplit

This menu command cancels horizontal or vertical splits of the window.

10.1.28 Window > Close all

This menu command closes all open windows.

10.1.29 Automatic model creation > Instantiate templates

Opens the "Instantiate templates" dialog box.

You can find additional information in the section Instantiation of templates from files or the simulation model (Page 286).

10.1.30 Automatic modelling > Automated import

Opens the file selection dialog to import models from the following sources:

- XML (Page 300)
- ZIP (Page 304)

You can find additional information on this in section: Import charts from ZIP or XML files (Page 299).

10.1.31 Automatic model creation > Bulk engineering import

Use this menu command to open the "Bulk engineering import" dialog box.

You can find additional information on this in section: "Bulk engineering import" dialog box (Page 996).

10.1.32 Options > Zoom

Use this menu command to zoom the selected object. Use the slider control to do this.

10.1.33 Options > SU administration

Use this menu command to open the "SU administration" dialog box:

For more information, visit:

- "SU administration" dialog box (Page 993)
- Configuring a SIMIT Unit (Page 94)
- Importing device description file to SIMIT (Page 95)
- Updating the firmware of the SIMIT Unit (Page 95)

10.1.34 Options > Assign coupling signals

With this menu command, the entries in input or output connectors are updated on the charts of the entire project, when the signal registered in the connector is clearly associated with a coupling.

Examples of applications:

- A coupling was subsequently renamed
- Coupling signals were subsequently assigned symbolic names

See also

Addressing a signal symbolically (Page 237)

10.1.35 Help > Display help

Use this menu command to access the SIMIT online help.

10.1.36 Help > About

This menu command displays the information about the SIMIT installation.

You can find additional information on this in the section: "Info" dialog box (Page 998).

10.1.37 Help > SIMIT resources

Use this menu command to call the "SIMIT Simulation Platform - Overview" page.

An active Internet connection is required.

10.1.38 Help > License container

Use this menu command to open an overview page in your default browser.

The overview page gives you information on which local license container you have access to and which licenses are included in it.

Information about licenses that are obtained via the network can be viewed in the "Diagnostics" menu item. You can reach the menu item "Diagnostics" via the start page of the CodeMeter WebAdmin in the browser.

An internet connection is not required for this.

10.1.39 CTE > Component > New component

You create a new component with this menu command.

10.1.40 CTE > Component > Open

You open an existing component with this menu command.

10.1.41 CTE > Component > Close

You close the open component with this menu command.

10.1.42 CTE > Component > Save

You save the current component with this menu command.

10.1.43 CTE > Component > Save as

You select an available storage location with this menu command where you save the current component.

10.1.44 CTE > Component > Exit

You exit the Component Type Editor with this menu command.

10.2 Dialog boxes

10.2.1 "Available backtracks" dialog box

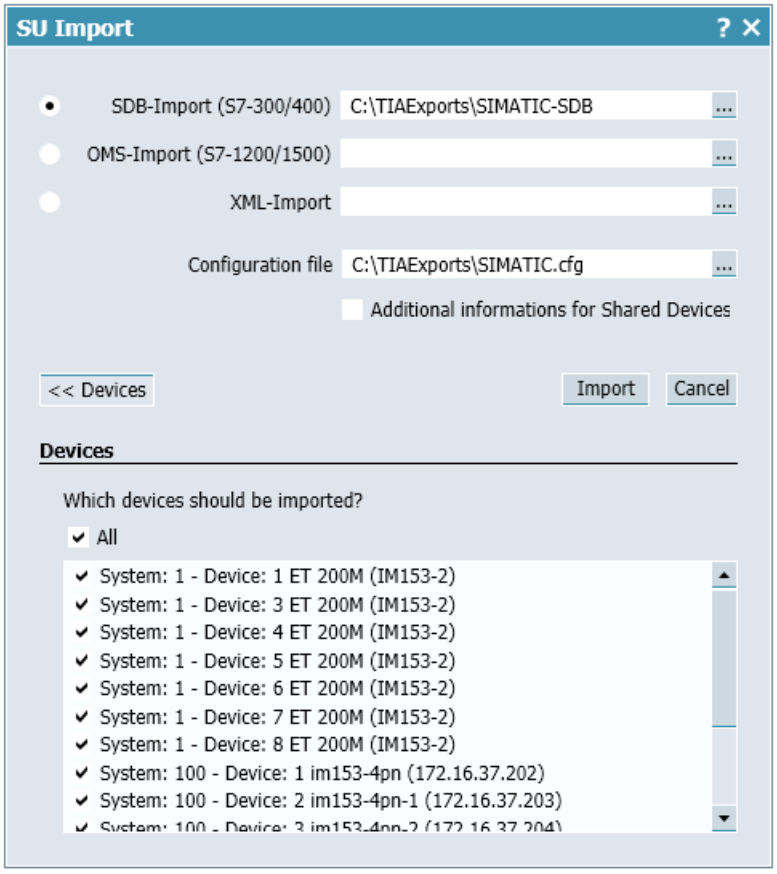
The dialog box opens when a running simulation is paused.

You execute the following functions in this dialog box:

- "Load backtrack"
  Loads a saved backtrack (state). The paused simulation jumps back to the selected point in time.
- "Convert to snapshot"
  Converts backtracks to regular snapshots.

You will find more information in section Backtracking. (Page 248)

10.2.2 "SU Import" dialog box



In this dialog box, you make the following settings for importing the hardware configuration:

- "SDB import"
  Defines the folder in which the system data blocks for importing the hardware configuration of an S7-300/S7-400 station are located.
- "OMS import"
  Defines the folder in which the system data blocks for importing the hardware configuration of a S7-1200/S7-1500 station are located.
  The OMS import is supported as of STEP 7 V13 and higher.
- "XML import"
  Specifies the file from which a vendor-neutral hardware configuration with PROFIBUS and/or PROFINET master systems is imported.
  You can find information on the structure of this file under "XML import interface (Page 116)".
- "Configuration file"
  Specifies the storage location of the configuration file with additional device information. Only relevant for "SDB import".
- "Additional information for Shared Devices"
  Specifies that the system data blocks of the existing hardware configuration of a station to be imported are completed. Only relevant for the simulation of a Shared Device.
- "Devices"
  Shows the area for station selection from the selected system files. All stations are selected by default.
- "Import"
  Starts station import to the SIMIT project.
- "Cancel"
  Cancels the import. Only the "SIMIT Unit" folder is created in the project tree.

10.2.3 "PLCSIM Advanced import" dialog box

In this dialog box, make the following settings for importing the hardware configuration of a STEP 7 project:

- TIA Portal project
  Specifies the project from which the hardware configuration is imported. These options are only available if PC STEP 7 in a compatible version is installed on the PC. See Compatibility tool (www.siemens.com/kompatool).
- HWCNExport file
  Specifies the HWCNExport file from which the hardware configuration is imported.

- Symbols
  Specifies how symbol names for inputs and outputs are treated. The selection is only in effect if symbol names were assigned in the STEP 7 project.

Symbol mode	Stations of the PLCSIM Advanced coupling are imported again (updated)	Stations of the PLCSIM Advanced coupling are created again
Create new	All symbols of the stations to be imported are deleted and the new symbols from the import are applied.	All symbols of the stations to be imported are applied from the import.
Overwrite	All symbols of the stations to be imported are applied from the import. Existing symbols are overwritten. New symbols are applied.	
Add	Only those symbols are applied from the import of the stations to be imported for which no prior symbols existed.	
Ignore	The symbols of the stations to be imported are not applied. Existing symbols remain unchanged.	The symbols of the stations to be imported are not applied.

- Adapt data width
  Specifies that the data width of the imported signals are matched to the symbols. If this option is not enabled, only symbols linked with the default data type Word will be applied.
- Stations >>
  Shows the area for station selection from the selected project. The stations are read in from the selected project. Only S7-1500 stations that are supported by PLCSIM Advanced are shown.
  All supported stations of the STEP 7 project are selected by default.
- Import
  Starts station import to the SIMIT project.
- Cancel
  Cancels the import. Only the "PLCSIM Advanced" folder is created in the project navigation.

See also

Creating a coupling of the "PLCSIM Advanced" type (Page 157)

Importing hardware configuration to station (Page 158)

10.2.4 "Virtual controller import" dialog box

This dialog box allows you to make the following settings to import the hardware configuration of a STEP 7/PCS 7 project:

- STEP 7/PCS 7 multiproject / STEP 7/PCS 7 project
  Specifies the project from which the hardware configuration is imported. These options are only available if SIMATIC Manager is installed.
- HWCNExport file
  Specifies the HWCNExport file from which the hardware configuration is to be imported.

- Update from Engineering
  Imports the hardware configuration of the selected engineering project from the engineering server.

  Note

  You will find more information in section Properties (Page 248).

  The function currently only supports PCS neo projects.

- Symbols
  Specifies how symbol names for inputs and outputs are treated. The selection is only effective if symbol names are assigned in the STEP 7/PCS 7 project.

Symbol mode	Virtual controller is imported again (updated)	Virtual controller is created
Create new	All symbols of the virtual controllers to be imported are deleted and the symbols from the import saved.	All symbols of the virtual controllers to be imported are taken from the import.
Overwrite	All symbols of the virtual controllers to be imported are taken from the import. Existing symbols are overwritten. New symbols are applied.	
Add	Symbols are only taken from the import for virtual controllers for which no symbols have yet been saved.	
Ignore	The symbols of the virtual controllers to be imported are not applied. Existing symbols remain unchanged.	The symbols of the virtual controllers to be imported are not applied.

- Adapt data width
  Specifies that the data width of the imported signals are matched to the symbols. If this option is not enabled, only symbols linked with the default data type Word will be applied.
- Stations >>
  Shows the area for station selection from the selected project. The stations are read in from the selected project. How long this process takes depends on the size of the STEP 7/PCS 7 project. It cannot be canceled.
  By default, all stations of an STEP 7/PCS 7 project are selected.
- Import
  Starts station import to the SIMIT project.
- Cancel
  Cancels the import. Only the "Virtual Controller" folder is created in the project tree.

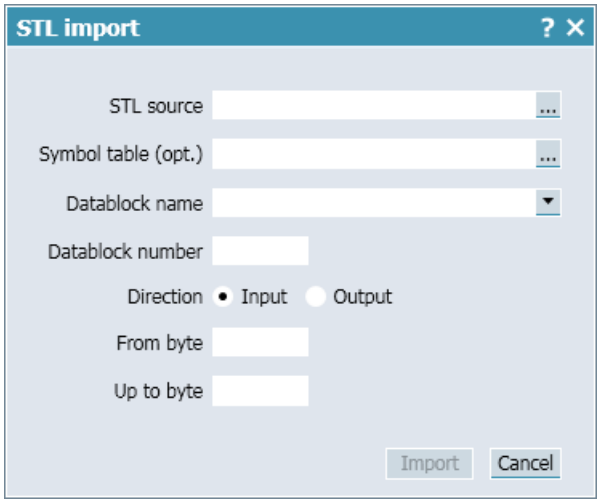
See also

HWCNExport introduction (Page 62)

Importing a STEP 7/PCS 7 project (Page 142)

Creating a Virtual Controller (Page 141)

10.2.5 "STL import" dialog box



Open the dialog box by clicking on the "STL import" button on the "DB" tab in the virtual controller coupling editor.

The following settings are made in this dialog box:

- "STL sources"
  Specifies the STL source
- "Symbol table (opt.)"
  Specifies the symbol table.
  If the data blocks in the STL source have symbolic names, the system can establish the data block number if you specify the symbol table. The numbers of the data blocks are read in from the symbol table and displayed in the dialog box under "Data block number".
- "Data block name"
  For import from a symbol table, specifies the data block from which a memory area is to imported.
- "Data block number"
  Specifies the data block number from which a memory area is to imported.
  Data block numbers entered manually are not validated. When you import from a symbol table, the data block number is already entered.
- "Direction"
  Specifies when input or output signals are imported.
- "Start byte"
  Specifies the start byte from which the signals are imported.

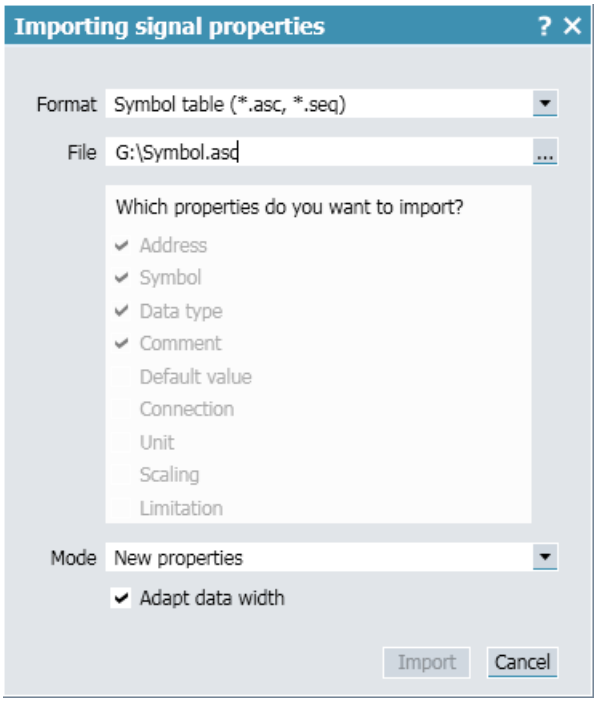
- "End byte"
  Specifies the end byte up to which the signals are imported. Here, you enter, at most, the end of the data block.
- "Import"
  Starts the import. The following fields must be filled out:
  - STL source
  - Data block number
  - Start byte
  - End byte

  Any existing signals in the specified memory area of "Start byte" and "End byte" are overwritten.

See also

Importing DB data from STL sources (Page 155)

10.2.6 "Importing signal properties" dialog box



In this dialog box, you make the following settings for importing signal properties:

- "Format"
  The format that can be read depends on the current coupling type. The following file formats are read in:
  - SIMATIC symbol table from STEP 7 or PCS 7
    You can find additional information in section: Symbol table (Page 222).
  - PLC tag table from the TIA Portal
    You can find additional information in section: Variable table (Page 223).
  - Signal table (txt file)
    You can find additional information in section: Signal table (Page 223).

    Note

    OPC coupling

    If signal properties were saved in txt format, the file also contains SIMATIC addresses and information for scaling. However, this information cannot be processed by an OPC coupling. If a txt file is imported into a OPC connection, the only information included is that which an OPC coupling can also process.

  - INI file
    You can find additional information in: INI format of the OPC couplings (Page 224)
- "File"
  Specifies the storage location of the import file.
- "**"Which properties do you want to import?"** area"
  Specifies which signal properties are to be imported. You can only import the signal properties that are in the import file. The following signal properties are imported by default when you import a symbol table or tag table:
  - Address
  - Symbol
  - Data type
  - Comment

- "Mode"
  The mode determines how the signals already present in the coupling and their properties are processed. The import modes that can be selected depends on the current coupling type.

  Note

  The import modes "New signals" and "Add signals" are supported by the following coupling types:
  - SHM
  - OPC
  - PLCSIM
  - PRODAVE

  The import modes "New properties", "Overwrite properties" and "Add properties" are supported by all coupling types.

  The following import modes are available:

  – New signals
    All existing signals are deleted and new signals are created with the imported information. The only properties imported are the once activated in the "Which properties do you want to import?" area. All other properties retain their default settings.

  – Add signals
    All signals from the import file are added. The only properties imported are the once activated in the "Which properties do you want to import?" area. All other properties retain their default settings. Already existing signals are not changed.

  – New properties
    The properties selected in the "Which properties do you want to import?" area are set to their default values for all signals available in the coupling and then taken from the import file.

  – Overwrite properties
    All properties selected in the "Which properties do you want to import?" area are taken from the import file and applied to the existing signals.

  – Add properties
    All properties selected in the "Which properties do you want to import?" area that still have their default settings are taken from the import file and applied to the existing signals.

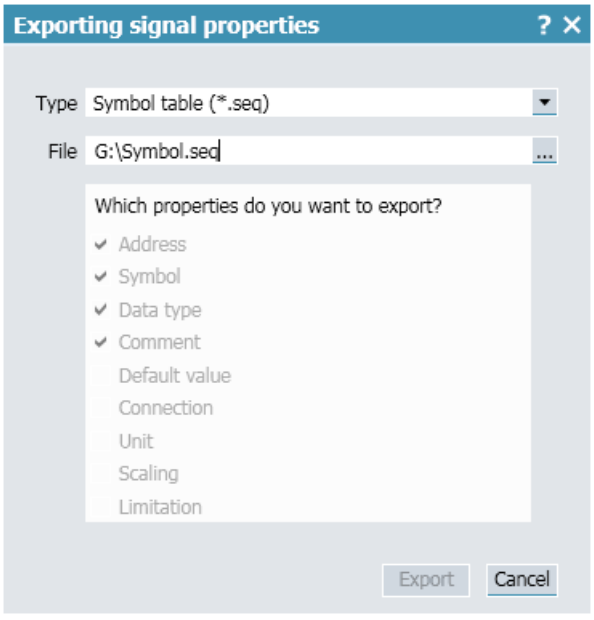
- **"**Adapt data width"

  Note

  This checkbox is only available for the following coupling types:
  - SIMIT Unit
  - Virtual controller
  - PLCSIM Advanced

  Specifies that the data width of signals is to be automatically adjusted to data width specified in the import file upon import. You can find additional information in section: Converting the data width of signals (Page 217).
  If the check box is not selected and the width of the data to be imported does not match the existing signal, this signal is not imported.

10.2.7 "Exporting signal properties" dialog box



In this dialog box, you make the following settings for exporting signal properties:

- "Type"
  Specifies the file format of the export file. The file format depends on the coupling type whose signal properties you are exporting.
  - SIMATIC symbol table from STEP 7/PCS 7 (seq file)
    This format only contains the signals from the coupling for which a symbolic name has been assigned.
    You can find additional information in section: Symbol table (Page 222).
  - Signal table (txt file)
    This format contains all signals from the coupling.
    You can find additional information in section: Signal table (Page 223).
  - INI format

    Note

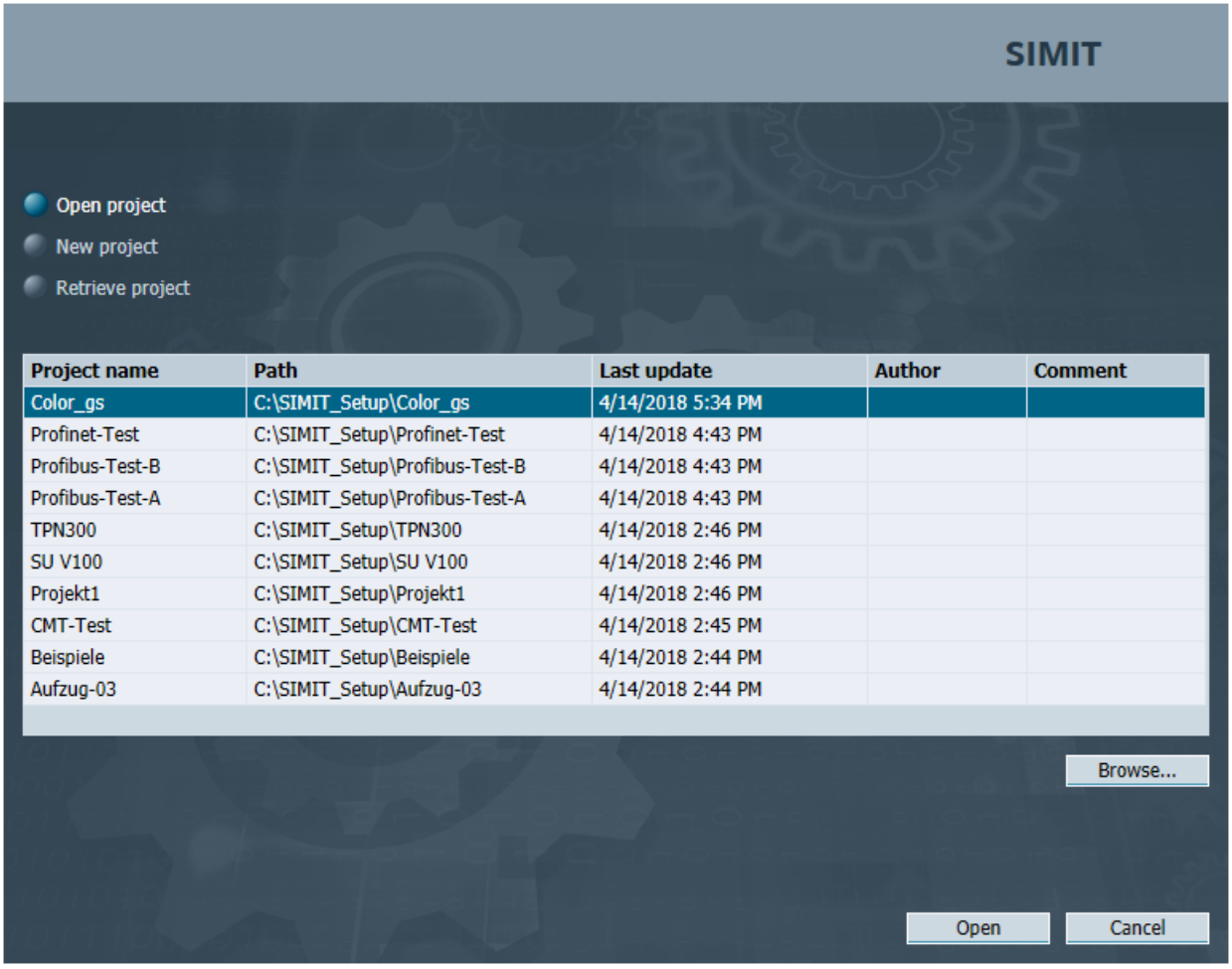
    OPC coupling

    Since OPC signals do not have an address nor can they be transmitted as scaled raw values, the signal table can only be filled to a limited extent. Such a signal table can be imported into another OPC coupling, but import into couplings that expect SIMATIC signals is not possible.

- "File"
  Specifies the storage location of the export file.

- "**"Which properties do you want to export?"** area"
  Specifies which signal properties are to be exported. The "Address" uniquely identifies a signal and is always exported. If you have selected the file format "Symbol table", the following signal properties are also exported:
  - Symbol
  - Data type
  - Comment
- "Export"
  Starts the export.

10.2.8 "Select project" dialog box



This dialog box is opened with the following menu commands:

- "Project > New project ..."
- "Project > Open"
- "Project > Retrieve"

The appearance of the dialog box varies depending on the selection.

The following functions are run in this dialog box:

- "Open project"
  Opens an existing project.
  You can find additional information on this in section: Project > Open... (Page 975). SIMIT projects have the file extension ".simit".
- "New project"
  Creates a new project.
  You can find additional information on this in section: Project > New project ... (Page 974).
- "Retrieve project"
  Retrieves an archived project.
  You can find additional information on this in section: Project > Retrieve (Page 977). Archived SIMIT projects have the file extension ".simarc".

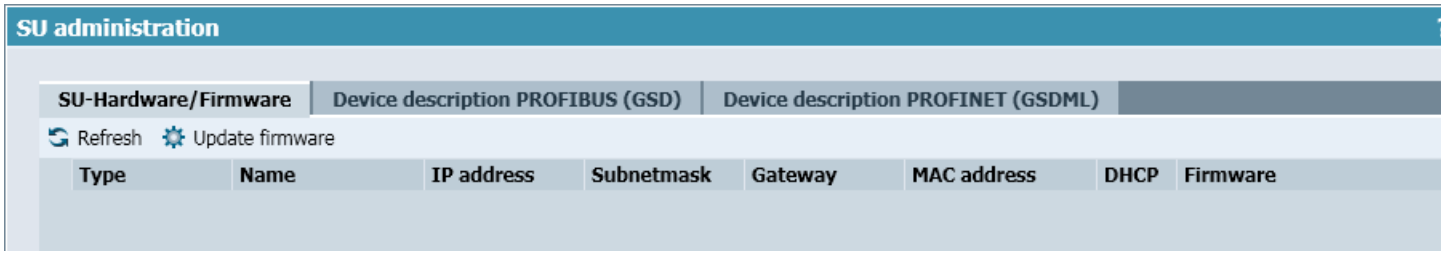
SIMIT projects can be stored in any folder in the file system. To select a folder, click "Browse...".

Note

SIMIT manages all information in this folder that is relevant for the project. Note the following:

- Do not rename this folder.
- Do not change any files in this folder.
- Do not save any files in this folder.
- Do not create any files in this folder.

10.2.9 "SU administration" dialog box



In this dialog box, you configure the SIMIT Unit.

"SU hardware/firmware" tab

This tab lists all SIMIT Units accessible in the network. You can find additional information in section Configuring a SIMIT Unit (Page 94).

- **Refresh**
  Refreshes the view.
- **Firmware update**
  Updates the firmware of the selected SIMIT Unit.

Via shortcut menu:

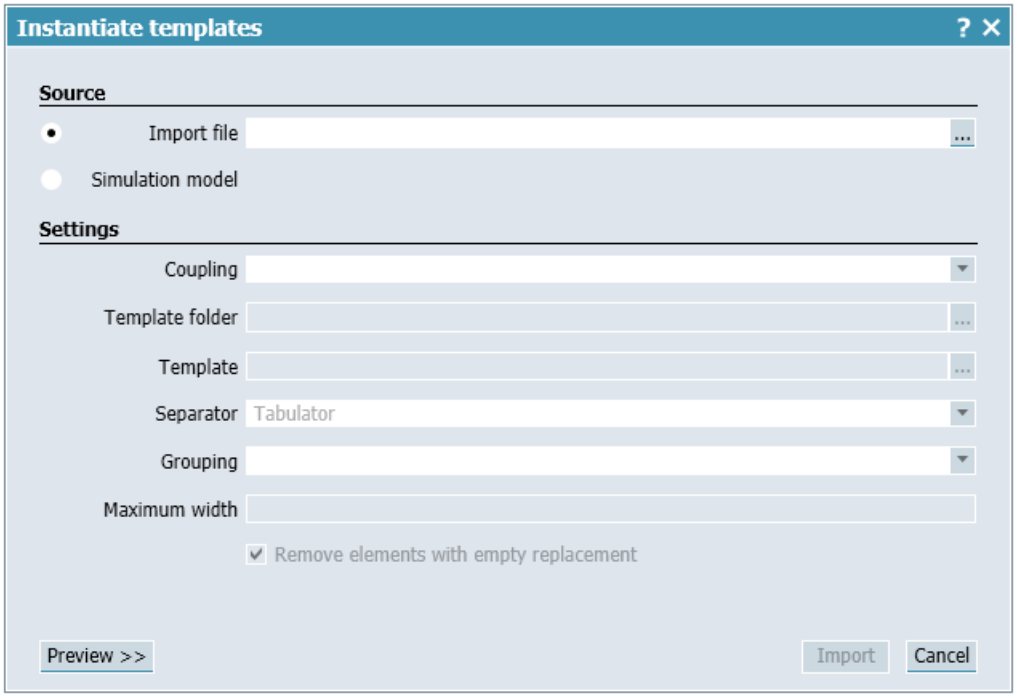
- **Firmware update**
  Updates the firmware of the selected SIMIT Unit.
- **Identify**
  Identifies the selected SIMIT unit by the flashing on the device.
- **Save log file**
  Loads the log files of the selected SIMIT unit.
  A dialog box opens for selecting the required files.
  A message indicating a successful download of the log files is displayed after the download is complete.
- **Reset**
  Resets the SIMIT Unit.
  This procedure corresponds to pressing the reset switch on the SIMIT Unit.

"Device description PROFIBUS (GSD)" and "Device description PROFINET (GSDML)" tabs

These tabs list the devices supported by the SIMIT Unit. You can expand the list of supported devices by importing device descriptions. You can find additional information in section Importing device description file to SIMIT (Page 95).

- **Refresh**
  Refreshes the view.
- **Import**
  Opens a dialog box for importing device descriptions from a device description file.
- **Delete**
  Removes a device imported from a device description file from the list.

10.2.10 "Instantiate templates" dialog box



You can make the following settings for instantiating a template in this dialog box:

- "Import file"
  Specifies that the templates are instantiated from a file.
  Specifies the import file. The following file types can be imported: *.txt, *.xls, *.xlsx, *.iea; *.xml.
- "Simulation model"
  Specifies that the templates are instantiated from the simulation model.
- "Coupling"
  Specifies the coupling.
- "Template folder"
  Specifies the folder in which the search for the SIMIT templates referenced in the CMT file is to take place.
  The name of the CMT type specified in the CMT file must exactly match the name of the template used in SIMIT.
- "Template"
  Specifies the storage location of the template.
- "Separator"
  Specifies the separator.
- "Grouping"
  Specifies the grouping.
- "Maximum width"/"Maximum height"
  Specifies the width/height in pixels.

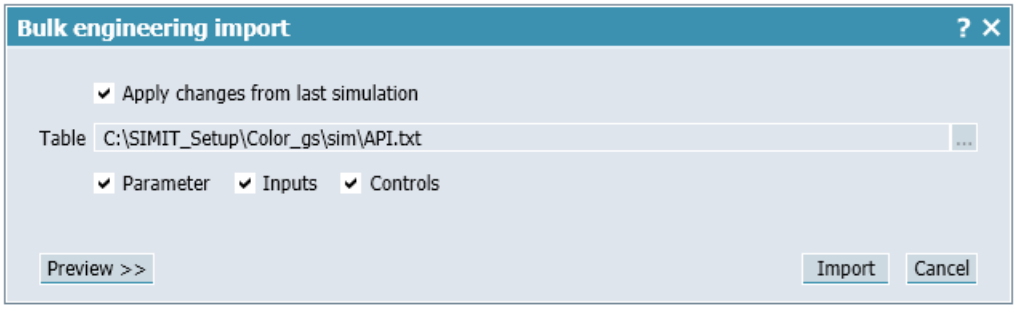
- "Remove elements with empty replacement"
  Specifies if empty replacements are to be removed from the file.
- "Preview >>"
  Shows a preview of the import.
  On the left in the preview, you can see the newly created folders and charts from the import. Individual folders and charts can be deselected. If you select a chart on the left, you will see the replacements to be performed on the right. The template used in each case is entered in the second header on the right and an indication is given if a template has been found at all.
- "Import"
  Starts the import.

See also

Instantiating templates from files (Page 286)

Instantiating templates from the simulation model (Page 298)

10.2.11 "Bulk engineering import" dialog box



The following settings for automatic parameter assignment and bulk data import are made in this dialog box:

- "Apply changes from last simulation"
  Specifies whether the changes from the most recently active simulation for this project are to be applied automatically.
  The check box is only enabled if at least one simulation has already been run for the project. Disable this option to import bulk data.
- "Table"
  Specifies the file from which the data is read.
- "Parameter"
  Specifies whether the parameters from the table are to be applied.
- "Inputs"
  Specifies whether the input defaults from the table are to be applied.
- "Controls"
  Specifies whether the default control settings from the table are to be applied.

- "Preview >>"
  Displays a preview of automatic parameter assignment.
- "Import"
  Runs automatic parameter assignment/Imports the modified values.

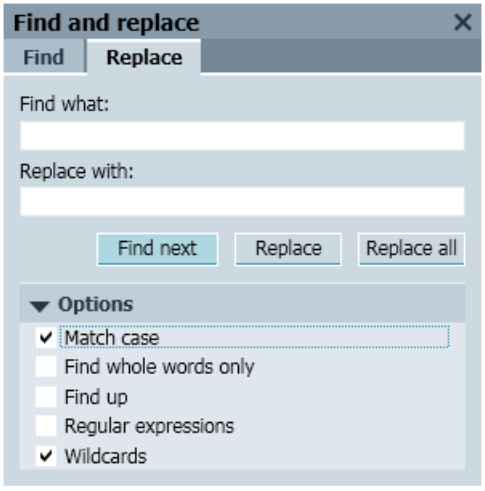
You can find additional information on this in section: Importing data (Page 307).

10.2.12 "Characteristic" editor

The characteristic editor is used to create a characteristic curve by specifying interpolation points and interpolation between these points.

You can find additional information in the section: Characteristic (Page 443)

10.2.13 "Find & replace" dialog box



Open the dialog box by pressing the shortcuts <Ctrl+F> for "Find" or <Ctrl+H> for "Replace". The search is limited to the area in which you opened the dialog box.

The following settings are made in this dialog box:

- "Find"
  Specifies the search term.
- "Find Next"
  Searches for the next occurrence of the search term.
- "Replace with"
  Specifies the term that will replace the search term.
- "Replace"
  Replaces the found search term.
  The "Find Next" function is automatically executed next.

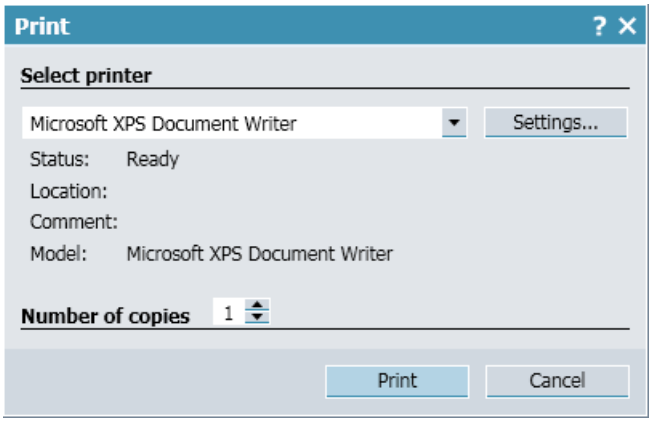
- "Replace all"
  Replaces all search terms found.
- "Options"
  Shows a configuration area in which you can enable additional options.
  Supported are, for example, "regular expressions" or "placeholders". The following characters can serve as placeholders:
  - "?": Represents a variable character. The search term "H?llo", for example, finds "Hello" and "Hallo"
  - "*": Represents any number of variable characters

10.2.14 "Info" dialog box

This menu command displays the following information about your SIMIT installation:

- Version
- Version of the SU software
- Version of SIMIT Virtual Controller (VC) (if installed)

10.2.15 "Print" dialog box



Select a printer from the drop-down list and set the number of copies.

Click the "Settings ..." button to change the settings of the selected printer.

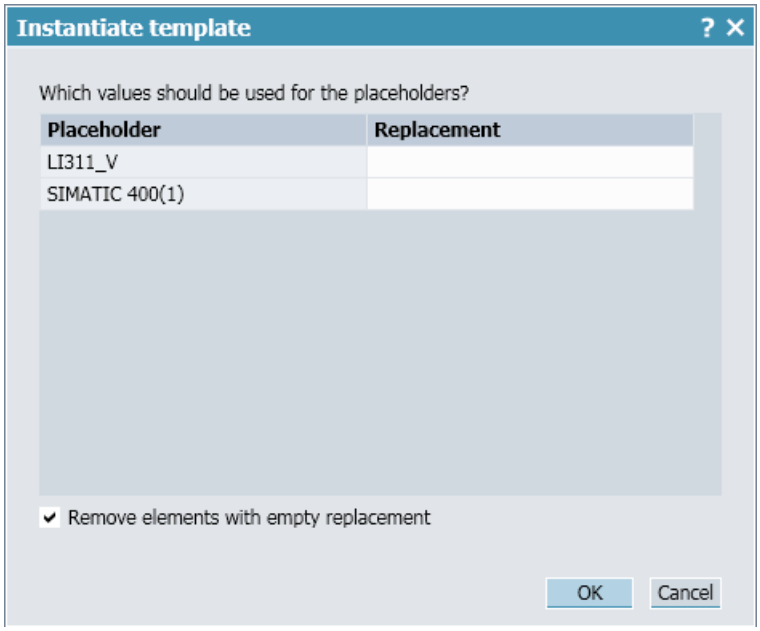
Click "Print" to start printing and close the dialog box.

See also

Printing charts (Page 258)

Printing trends (Page 322)

10.2.16 "Instantiate template" dialog box



In this dialog box, the parameters of the selected template are listed in the "Placeholder" column. Arbitrary values can be entered as replacements in the "Replacement" column.

Select the "Remove elements with empty replacement" check box to remove parameter in the template for which you have not entered a name in the "Replacement" column.

Click "OK" to apply the changes and close the dialog box.

10.2.17 "Selection" dialog box

Select a new coupling in this dialog box. The following couplings are available for selection:

Hardware

- **SIMIT Unit**
  You can find more information on the coupling under SIMIT Unit (Page 65).
- **PRODAVE**
  You can find more information on the coupling under PRODAVE coupling (Page 200).

Standard

- **OPC DA Client**
  You can find more information on the coupling under OPC coupling (Page 165).
- **OPC DA Server**
  You can find more information on the coupling under OPC coupling (Page 165).
- **OPC UA Client**
  You can find more information on the coupling under OPC coupling (Page 165).

- **OPC UA Server**
  You can find more information on the coupling under OPC coupling (Page 165).
- **Shared Memory**
  You can find more information on the coupling under Shared memory coupling (Page 191).

Additional

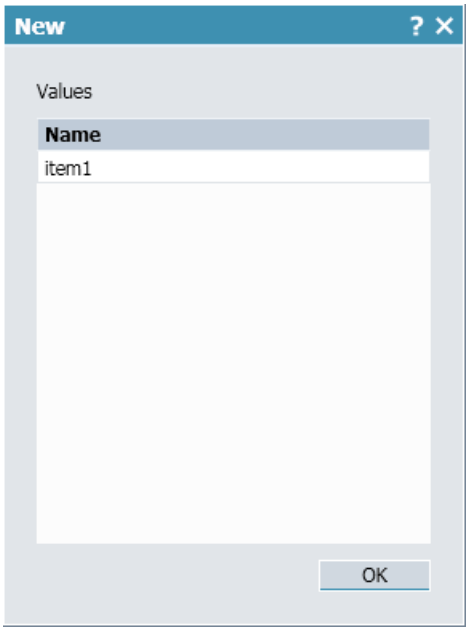
- **gPROMS**
  You can find more information on the coupling under OPC coupling (Page 165).
- **Mechatronics Concept Designer**
  You can find more information on the coupling under Mechatronics Concept Designer coupling (Page 203).
- **Plant Simulation**
  You can find more information on the coupling under Plant Simulation (Page 200).
- **TableReader**
  You can find more information on the coupling under TableReader coupling (Page 244).

Emulation

- **Virtual Controller**
  You can find more information on the coupling under Virtual Controller (Page 122).
- **PLCSIM Advanced**
  You can find more information on the coupling under PLCSIM Advanced coupling (Page 157).
- **PLCSIM**
  You can find more information on the coupling under PLCSIM coupling (Page 163).

Click "OK" to confirm the selection and close the dialog box.

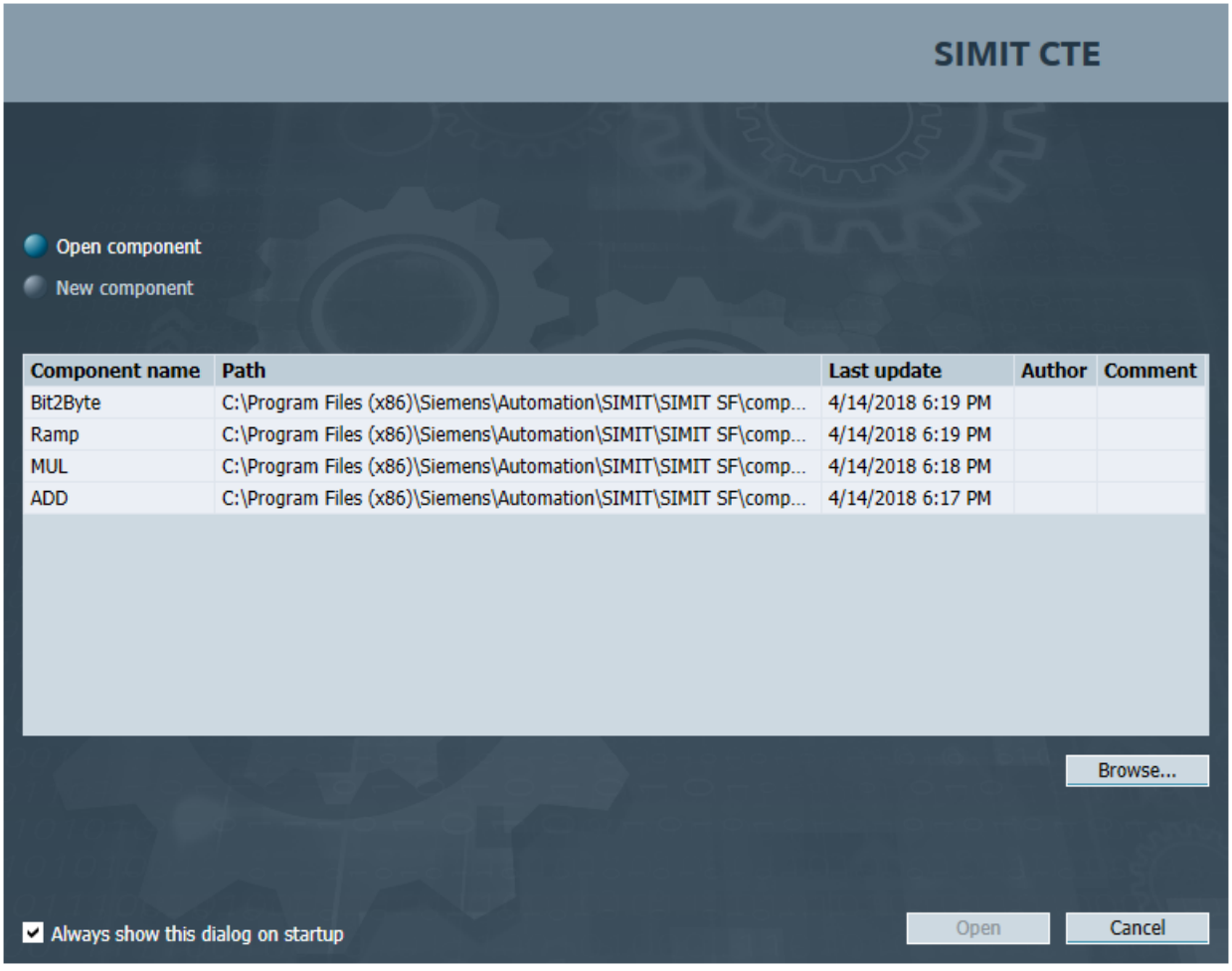
10.2.18 "Enumeration" dialog box (CTE)



You can enter the names for elements of custom enumeration type in this dialog box.

You can find additional information in the section: The "Enumeration Types" task card (Page 362).

10.2.19 "Manage components" dialog box (CTE)



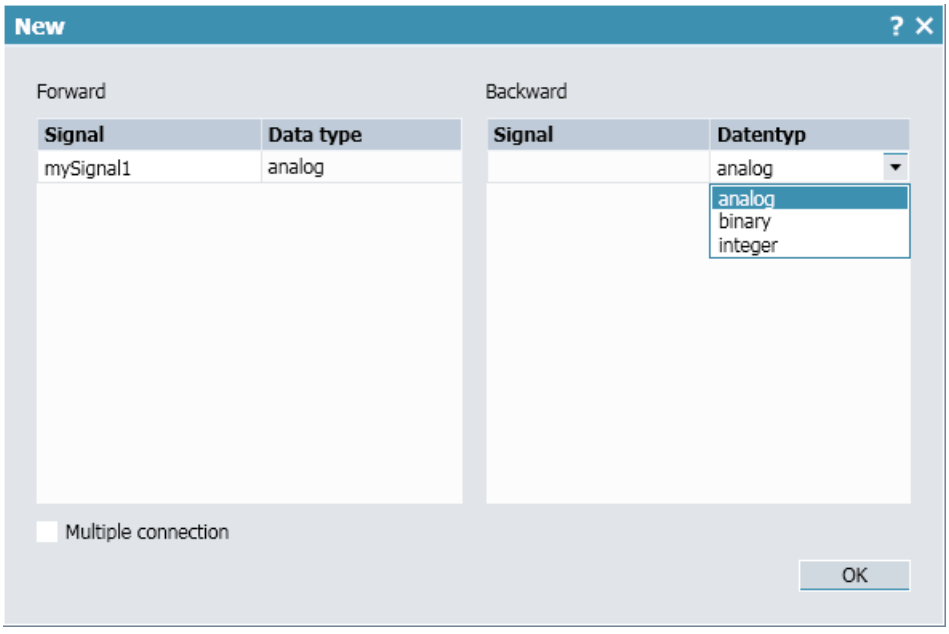
You can create new components or open existing ones in this dialog box. To do this, select the desired function.

Click the "Browse..." button to select the storage location of the component.

Click the "Open" button to open the component in the editor.

Select the check box "Always show this dialog on startup", if this dialog box is to open automatically when you start the CTE.

10.2.20 "Connection type" dialog box (CTE)



You define the signals of custom connection type in this dialog box. You select the data type of the signal from the drop-down list. You can create any number of signals.

Select the "Multiple connection" check box if an output is to be connected to more than one input.

You can find more information in section: The "Connection Types" task card (Page 363).