SIEMENS

# SIMATIC

# Process Control System PCS 7 Advanced Process Library (V9.1 SP1)

Function Manual

11/2021
A5E48022986-AB

Security information 1
Basics of APL 2
Operator control blocks 3
Condition Monitoring 4
Monitoring blocks 5
Controller blocks 6
Dosing blocks 7
Motor and valve blocks 8
Interlock blocks 9
Message blocks 10
Counter blocks 11
Timers 12
Mathematical blocks 13
Analog logic blocks 14
Digital logic blocks 15
Generator blocks 16
Channel blocks 17
Conversion blocks 18
Maintenance blocks 19
System blocks 20
Communication blocks 21
Process tag types (insertible templates) 22
Definitions 23

## 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

A5E48022986-AB
Ⓟ 12/2021 Subject to change

Copyright © Siemens AG 2021.
All rights reserved

# Table of contents


**1 Security information ..... 43**

**2 Basics of APL ..... 45**

2.1 Functions of the blocks ..... 45
2.1.1 General information ..... 45
2.1.1.1 User-configured message classes ..... 45
2.1.1.2 Forcing operating modes ..... 45
2.1.1.3 Resetting the block in case of interlocks or errors ..... 47
2.1.1.4 Neutral position for motors, valves and controllers ..... 54
2.1.1.5 Specifying warning times for control functions at motors and valves ..... 56
2.1.1.6 Output signal as a static signal or pulse signal ..... 57
2.1.1.7 Recording the first signal for interlock blocks ..... 58
2.1.1.8 Outputting a signal for start readiness ..... 59
2.1.1.9 Simulating signals ..... 64
2.1.1.10 Dead band ..... 67
2.1.1.11 Request for maintenance release ..... 68
2.1.1.12 Release for maintenance ..... 70
2.1.1.13 SIMATIC BATCH functionality ..... 73
2.1.1.14 Flutter suppression for channel blocks ..... 73
2.1.1.15 Startup characteristics over Trigger block ..... 75
2.1.1.16 Parameters that cannot be connected ..... 75
2.1.2 Operating modes of the blocks ..... 76
2.1.2.1 Overview of the modes ..... 76
2.1.2.2 On ..... 77
2.1.2.3 Out of service ..... 77
2.1.2.4 Manual and automatic mode for control blocks ..... 79
2.1.2.5 Manual and automatic mode for motors, valves and dosers ..... 82
2.1.2.6 Program mode for controllers ..... 85
2.1.2.7 Local mode ..... 86
2.1.2.8 State graph of the operating modes ..... 90
2.1.3 Monitoring functions ..... 92
2.1.3.1 Monitoring functions in the Advanced Process Library ..... 92
2.1.3.2 Group display for limit monitoring, CSF and ExtMsgx ..... 92
2.1.3.3 Limit monitoring ..... 93
2.1.3.4 Feedbacks ..... 104
2.1.3.5 Motor protection function ..... 106
2.1.4 Interlocking functions ..... 107
2.1.4.1 Interlocks ..... 107
2.1.4.2 Disabling interlocks ..... 111
2.1.4.3 Influence of the signal status on the interlock ..... 112
2.1.4.4 Forming the group status for interlock information ..... 113
2.1.4.5 Rapid stop for motors ..... 115
2.1.4.6 Bypassing signals ..... 115
2.1.5 Form signal status ..... 116
2.1.5.1 Forming and outputting signal status for blocks ..... 116
2.1.5.2 Forming and outputting the signal status for technologic blocks ..... 117

2.1.5.3 Forming and outputting the signal status of digital logic blocks ..... 119
2.1.5.4 Forming and outputting the signal status of analog logic blocks ..... 120
2.1.5.5 Forming and outputting the signal status of redundancy blocks ..... 121
2.1.5.6 Forming and outputting the signal status for blocks with configurable status prioritization ..... 122
2.1.5.7 Forming and outputting the signal status for interlock blocks ..... 124
2.1.5.8 Forming and outputting the signal status for mathematical blocks ..... 126
2.1.5.9 Forming and outputting the signal status for PCS 7 channel blocks ..... 127
2.1.5.10 Forming and outputting the signal status for channel blocks for field devices ..... 127
2.1.6 Error handling ..... 128
2.1.6.1 Error handling ..... 128
2.1.6.2 Outputting group errors ..... 130
2.1.7 Ramp function ..... 131
2.1.7.1 Using setpoint ramp ..... 131
2.1.7.2 Gradient limit of the setpoint ..... 132
2.1.7.3 Using a manipulated variable ramp ..... 133
2.1.7.4 Gradient limiting of the manipulated variable ..... 134
2.1.8 Internal/external setting ..... 135
2.1.8.1 Applying the dynamically activated dead band during the PV settling time ..... 135
2.1.8.2 Setpoint specification - internal/external ..... 136
2.1.8.3 Manipulated variable specification - internal/external ..... 137
2.1.9 Configurable response using the Feature I/O ..... 139
2.1.9.1 Configurable functions using the Feature I/O ..... 139
2.1.9.2 Stopping dosing at a flow alarm ..... 147
2.1.9.3 Failure handling ..... 148
2.1.9.4 Set startup characteristics ..... 148
2.1.9.5 Applying the dynamically activated dead band during the PV settling time ..... 152
2.1.9.6 Evaluation of signal status ..... 152
2.1.9.7 Evaluation of the signal status of the interlock signals ..... 153
2.1.9.8 Automatic post dosing for underdosing in automatic mode ..... 153
2.1.9.9 Block as summing unit or integrator ..... 154
2.1.9.10 Switching operator controls for external setpoint to visible ..... 154
2.1.9.11 Limit output Out ..... 154
2.1.9.12 Activating calculation of the flow rate for dosing by scale ..... 155
2.1.9.13 Condition monitoring information at MOD_Blocks ..... 155
2.1.9.14 Condition monitoring information at the channel blocks ..... 155
2.1.9.15 Disabling operating points ..... 156
2.1.9.16 Enabling direct changeover between forward and reverse ..... 156
2.1.9.17 Specifying the dosing type ..... 157
2.1.9.18 Flow setpoints in percent ..... 157
2.1.9.19 Specifying the influence of the signal status on the dosing process ..... 158
2.1.9.20 Unit for the rate of change ..... 158
2.1.9.21 Reading messages ..... 158
2.1.9.22 Setting the scaling for the process values ..... 159
2.1.9.23 Outputting a de-energized value for block-external simulation ..... 159
2.1.9.24 Switch to substitute value ..... 159
2.1.9.25 Substitute value switch in the event of an error ..... 160
2.1.9.26 Output substitute value if raw value is invalid ..... 160
2.1.9.27 Activating recording of the first signal ..... 161
2.1.9.28 External control deviation ..... 161
2.1.9.29 Activating error state for external process control error CSF ..... 162
2.1.9.30 Frequency converter with separate device feed ..... 162
2.1.9.31 Separate evaluation for excluded and simulated interlock signals ..... 162

2.1.9.32 Use an internal or external setpoint for the absolute fine dosing quantity ..... 164
2.1.9.33 Activating the run time of feedback signals ..... 164
2.1.9.34 Outputting last valid value if raw value is invalid ..... 165
2.1.9.35 Use the last value following a complete download as the current value during startup of the block ..... 165
2.1.9.36 Activate LowCutOff ..... 166
2.1.9.37 Selecting values associated with messages ..... 167
2.1.9.38 Reporting with BATCH parameters ..... 167
2.1.9.39 Motor feedback is not available ..... 168
2.1.9.40 Display only input values that are interconnected in the faceplate ..... 168
2.1.9.41 Activate OS_Perm bits ..... 168
2.1.9.42 Disabling opening and closing ..... 169
2.1.9.43 Enabling local operator authorization ..... 169
2.1.9.44 Enable configuration of the dribbling quantity ..... 170
2.1.9.45 Enabling program mode ..... 170
2.1.9.46 Update acknowledgment and error status of the message call ..... 171
2.1.9.47 Control zone with frozen I component ..... 171
2.1.9.48 Control zone with specified I component ..... 171
2.1.9.49 Resetting the commands for changing the mode ..... 172
2.1.9.50 Enabling resetting of commands for the control settings ..... 172
2.1.9.51 Resetting the dosing quantity when dosing starts ..... 173
2.1.9.52 Process value with separate scale range ..... 173
2.1.9.53 Resetting via input signals in the event of interlocking (Protection) or errors ..... 174
2.1.9.54 Set reset depending on the operating mode or the LiOp parameter ..... 174
2.1.9.55 Activating reset of protection / error in manual mode ..... 175
2.1.9.56 Reset even with locked state ..... 176
2.1.9.57 Neutral position manipulated variable takes effect at startup ..... 176
2.1.9.58 Neutral position manipulated variable takes effect with "out of service" operating mode ..... 177
2.1.9.59 Setting switch or button mode ..... 177
2.1.9.60 Specifying switching mode ..... 178
2.1.9.61 Creep rate is always detected in the dosing quantity ..... 179
2.1.9.62 Enabling rapid stop via faceplate ..... 179
2.1.9.63 Position feedback signals are active ..... 179
2.1.9.64 Separate monitoring time for stopping the motor ..... 180
2.1.9.65 Separate interlock for each direction, position, or speed ..... 180
2.1.9.66 Separate delay times for each alarm ..... 181
2.1.9.67 Signaling limit violation ..... 181
2.1.9.68 Alarm setpoint difference ..... 182
2.1.9.69 Setpoint specification with separate display area and custom unit ..... 182
2.1.9.70 Control via auxiliary valve ..... 182
2.1.9.71 Enabling bumpless change to the proportional gain, derivative time and amplification of the differentiator ..... 183
2.1.9.72 Enable bumpless switchover to "Automatic" mode for operator only ..... 183
2.1.9.73 Disabling bumpless switchover to automatic mode for controllers ..... 183
2.1.9.74 Enabling bumpless switchover to automatic mode for valves, motors, and dosers ..... 184
2.1.9.75 Summing characteristics continuous or triggered ..... 184
2.1.9.76 Suppression of all messages ..... 185
2.1.9.77 Output invalid raw value ..... 185
2.1.9.78 Transmission of status information of devices ..... 186
2.1.9.79 Control priority in the event of invalid or not plausible input commands ..... 186
2.1.9.80 Sealing the valve ..... 187
2.1.9.81 First-in detection response to deactivation ..... 187

2.1.9.82 Reaction of the switching points in the "Out of service" operating mode ..... 188
2.1.9.83 Reaction to the out of service mode ..... 188
2.1.9.84 Exiting local mode ..... 189
2.1.9.85 Interlock display with LocalSetting 2 or 4 ..... 189
2.1.9.86 Vibrate after torque monitoring ..... 190
2.1.9.87 With acknowledge overdosage ..... 190
2.1.9.88 Ramp rate calculation ..... 191
2.1.9.89 SP following PV in open loop has no priority over SP_Ext and SP limits ..... 193
2.1.9.90 With accelerated return of the integral action from the manipulated variable limit ..... 193
2.1.9.91 External/internal selection specification ..... 194
2.1.9.92 Setting switch or button mode for local commands ..... 195
2.1.9.93 Gradient limitation with time duration ..... 197
2.1.9.94 Analog input 1 is reserved for the operator ..... 197
2.1.9.95 Operator can change the setpoint via faceplate also in the "Local" mode ..... 197
2.1.9.96 Define the setpoint after stop and start of the motor ..... 198
2.1.9.97 Retain last output value in case of bad input signal status ..... 198
2.1.9.98 Forcing commands in the Local mode ..... 198
2.1.9.99 Motor stop in end position depends only on the corresponding feedback signal ..... 199
2.1.9.100 Substitution value is active if the block is in bypass ..... 199
2.1.9.101 Enable external message ..... 200
2.1.9.102 Considering bad quality of automatic commands or external values ..... 200
2.1.9.103 Disable calculation of impulse control in LocalSetting 2 and 4 ..... 200
2.1.9.104 Inverter enable ..... 201
2.1.9.105 Scale dosing can be started only if StandStill = 1 ..... 201
2.1.9.106 Delete history of MV and DV moves during restart ..... 201
2.1.9.107 Status output calculation ..... 202
2.1.9.108 Suppress MsgLock and “Out of service” mode for a connected message block ..... 202
2.1.9.109 Variants of CPI calculation ..... 202
2.1.9.110 Scalable measuring range ..... 203
2.1.9.111 2 decimal points (scalable measuring range) ..... 203
2.1.9.112 3 decimal points (scalable measuring range) ..... 204
2.1.9.113 Output substitute values in case of communications error ..... 204
2.1.9.114 Output last values in case of communications error ..... 204
2.1.9.115 Dosing controls are deactivated during scale is in taring mode ..... 205
2.1.9.116 Do not evaluate the first run of the diagnostic blocks ..... 205
2.1.9.117 Selection of metadata and diagnostic blocks ..... 205
2.1.9.118 Limits disabled in faceplate ..... 206
2.1.9.119 Keep last valid raw value if input value is bad ..... 207
2.1.9.120 With PID initialization if mode SP intern extern changes ..... 207
2.1.9.121 Changes signal status of outputs in OOS ..... 208
2.1.9.122 Behavior in the case of Bus failure ..... 208
2.1.9.123 Button mode with triggered local control commands ..... 209
2.1.9.124 Understeering / underflow for measuring range 0...10V ..... 209
2.1.9.125 Disable limit act output when limit value message is disabled ..... 209
2.1.9.126 Resetting the Pending Error in Local Mode 1 and 3 ..... 210
2.1.9.127 Timestamp calculation ..... 210
2.1.9.128 Reversal of control direction for fail-open valves ..... 211
2.1.9.129 External Input to enable or Disable ..... 211
2.1.9.130 Switch over to internal time enable or Disable ..... 211
2.1.9.131 AS time enable or Disable ..... 212
2.1.9.132 Use connectable inputs OpenLi, StopLi, CloseLi with CtrlLiOp ..... 212
2.1.9.133 Selecting the Feature Bit when Namur Off = 1 ..... 212

2.1.9.134 Bypass over both connection BypLix OR Inx ... 212
2.1.9.135 Resynchronization after startup ... 213
2.1.9.136 Resynchronization after out of service or after simulation ... 213
2.1.9.137 Resynchronization after CSF ... 214
2.1.9.138 Resynchronization with StateTrkOn input ... 214
2.1.9.139 Control with output StopCtrl and state "No Cmd" ... 214
2.1.10 Functions for controllers ... 216
2.1.10.1 Delay alarm for control deviation at setpoint step changes ... 216
2.1.10.2 Inverting control direction ... 217
2.1.10.3 Error signal generation and dead band ... 217
2.1.10.4 Using control zones ... 220
2.1.10.5 Setpoint limiting for external setpoints ... 222
2.1.10.6 Tracking setpoint in manual mode ... 222
2.1.10.7 Tracking and limiting a manipulated variable ... 222
2.1.10.8 Feedforwarding and limiting disturbance variables ... 223
2.1.10.9 Structure segmentation at controllers ... 224
2.1.11 Messaging ... 225
2.1.11.1 Area of application of the alarm delays ... 225
2.1.11.2 One time value for all limits ... 226
2.1.11.3 One time value per limit pair ... 226
2.1.11.4 Two time values per limit pair ... 227
2.1.11.5 Two time values for each individual limit ... 228
2.1.11.6 Generating instance-specific messages ... 230
2.1.11.7 Suppressing messages using the MsgLock parameter ... 231
2.1.11.8 Time stamp ... 231
2.1.11.9 Connection of the time-stamped messages from EventTs or Event16Ts ... 233
2.1.12 Settings for operator control and monitoring ... 233
2.1.12.1 Display and operator input area for process values and setpoints ... 233
2.1.12.2 Opening additional faceplates ... 234
2.1.12.3 Labeling of buttons and text ... 235
2.1.12.4 Displaying auxiliary values ... 239
2.1.12.5 Selecting a unit of measure ... 240
2.1.13 Consumption of Process Objects ... 258
2.1.14 Resynchronization ... 258

2.2 Functions of the block icons ... 260
2.2.1 Block icon structure ... 260
2.2.2 Configuring the block icons ... 267
2.2.3 Operation via the block icon ... 269
2.2.4 Block icons for PID and FM controller ... 269
2.2.5 Block icon for interlock blocks ... 272
2.2.6 Block icons for SFC ... 274
2.2.7 Adding block icons to static picture components ... 276
2.2.8 Decimal points for the analog value display ... 277

2.3 Functions of the faceplates ... 279
2.3.1 Fast Operation for PID Controllers ... 279
2.3.2 Structure of the faceplate ... 279
2.3.3 Operator control permissions ... 285
2.3.4 Display of delay times ... 288
2.3.5 Switching operating states and operating modes ... 289
2.3.6 Changing values ... 291
2.3.7 FM controllers standard view (analog) ... 294

2.3.8 FM controllers standard view (pulse controller) ... 298
2.3.9 FM controllers standard view (step controller with position feedback) ... 302
2.3.10 FM controllers standard view (step controller without position feedback) ... 306
2.3.11 Interlock blocks standard view ... 310
2.3.12 Parameter view of PID controllers ... 315
2.3.13 Parameter view of FM controllers ... 318
2.3.14 Parameter view for motors and valves ... 320
2.3.15 Limit view of FM controllers ... 322
2.3.16 Limit view of PID controllers ... 325
2.3.17 Limit view of motors ... 328
2.3.18 Preview of FM controllers ... 331
2.3.19 Preview of interlock blocks ... 333
2.3.20 Ramp view ... 334
2.3.21 Alarm view ... 336
2.3.22 Batch view ... 336
2.3.23 Memo view ... 338
2.3.24 Trend view ... 341
2.3.25 AddOn View for function blocks ... 343
2.3.26 APL Operator Trend Control (AOTC) ... 343
2.3.27 Limit operation and display in the faceplate ... 355
2.3.28 Central color management ... 356
2.3.29 Bar graph ... 376
2.3.30 Reset of errors during Request 0/1 ... 382

2.4 PCS 7 measuring point browser ... 383
2.4.1 Overview of the "PCS 7 measuring point browser" window ... 383

**3 Operator control blocks ... 387**

3.1 Comparison of large & small blocks ... 387
3.1.1 OpAnL compared to OpAnS ... 387

3.2 OpAnL - Check and output analog signals (large) ... 389
3.2.1 Description of OpAnL ... 389
3.2.2 OpAnL modes ... 390
3.2.3 OpAnL functions ... 391
3.2.4 OpAnL error handling ... 394
3.2.5 OpAnL messaging ... 395
3.2.6 OpAnL I/Os ... 397
3.2.7 OpAnL block diagram ... 403
3.2.8 Operator control and monitoring ... 403
3.2.8.1 OpAnL views ... 403
3.2.8.2 OpAnL standard view ... 404
3.2.8.3 OpAnL parameter view ... 407
3.2.8.4 OpAnL trend view ... 407
3.2.8.5 OpAnL preview ... 409
3.2.8.6 Block icon for OpAnL ... 410

3.3 OpAnS - Check and output analog signals (small) ... 412
3.3.1 Description of OpAnS ... 412
3.3.2 OpAnS modes ... 413
3.3.3 OpAnS functions ... 414
3.3.4 OpAnS error handling ... 416
3.3.5 OpAnS messaging ... 417
3.3.6 OpAnS I/Os ... 417

3.3.7 OpAnS block diagram ... 420
3.3.8 Operator control and monitoring ... 421
3.3.8.1 OpAnS views ... 421
3.3.8.2 OpAnS standard view ... 421
3.3.8.3 OpAnS parameter view ... 423
3.3.8.4 OpAnS preview ... 424
3.3.8.5 OpAnS block icon ... 425

3.4 OpDi01 - Manipulating a digital value (2 pushbuttons) ... 426
3.4.1 Description of OpDi01 ... 426
3.4.2 OpDi01 modes ... 427
3.4.3 OpDi01 functions ... 428
3.4.4 OpDi01 error handling ... 431
3.4.5 OpDi01 messaging ... 431
3.4.6 OpDi01 I/Os ... 432
3.4.7 OpDi01 block diagram ... 435
3.4.8 Operator control and monitoring ... 435
3.4.8.1 OpDi01 views ... 435
3.4.8.2 OpDi01 standard view ... 436
3.4.8.3 OpDi01 preview ... 437
3.4.8.4 Block icon for OpDi01 ... 439

3.5 OpDi03 - Manipulating a digital value (3 pushbuttons) ... 441
3.5.1 Description of OpDi03 ... 441
3.5.2 OpDi03 modes ... 442
3.5.3 OpDi03 functions ... 443
3.5.4 OpDi03 error handling ... 446
3.5.5 OpDi03 messaging ... 447
3.5.6 OpDi03 I/Os ... 448
3.5.7 OpDi03 block diagram ... 451
3.5.8 Operator control and monitoring ... 451
3.5.8.1 OpDi03 view ... 451
3.5.8.2 OpDi03 standard view ... 452
3.5.8.3 OpDi03 preview ... 454
3.5.8.4 Block icon for OpDi03 ... 455

3.6 OpStations - Configuration of the local operator authorization ... 457
3.6.1 Description of OpStations ... 457
3.6.2 OpStations operating modes ... 459
3.6.3 OpStations functions ... 460
3.6.4 OpStations error handling ... 461
3.6.5 OpStations messaging ... 461
3.6.6 OpStations I/Os ... 462
3.6.7 OpStations block diagram ... 464
3.6.8 Operator control and monitoring ... 466
3.6.8.1 OpStations views ... 466
3.6.8.2 OpStations standard view ... 467
3.6.8.3 Block icon of OpStations ... 469

3.7 OpTrig - Manipulating a digital value (1 pushbutton) ... 470
3.7.1 Description of OpTrig ... 470
3.7.2 OpTrig modes ... 471
3.7.3 OpTrig functions ... 471
3.7.4 OpTrig error handling ... 473

3.7.5 OpTrig messaging ..... 474
3.7.6 OpTrig I/Os ..... 474
3.7.7 OpTrig block diagram ..... 477
3.7.8 Operator control and monitoring ..... 477
3.7.8.1 OpTrig views ..... 477
3.7.8.2 OpTrig standard view ..... 478
3.7.8.3 OpTrig preview ..... 479
3.7.8.4 Block icon for OpTrig ..... 480

**4 Condition Monitoring ..... 483**

4.1 PumpMon - Pump monitoring ..... 483
4.1.1 Description of PumpMon ..... 483
4.1.2 PumpMon modes ..... 488
4.1.3 PumpMon functional description ..... 488
4.1.4 PumpMon Functions ..... 495
4.1.5 PumpMon error handling ..... 502
4.1.6 PumpMon messaging ..... 503
4.1.7 PumpMon I/Os ..... 505
4.1.8 PumpMon block diagram ..... 518
4.1.9 PumpMon additional information ..... 519
4.1.10 Operator control and monitoring ..... 521
4.1.10.1 Views of PumpMon ..... 521
4.1.10.2 PumpMon standard view (delivery height characteristic) ..... 522
4.1.10.3 PumpMon efficiency and power characteristic view ..... 524
4.1.10.4 PumpMon NPSH characteristic view ..... 525
4.1.10.5 PumpMon histogram view ..... 526
4.1.10.6 PumpMon parameter view ..... 529
4.1.10.7 PumpMon characteristic parameter view 1 ..... 531
4.1.10.8 PumpMon characteristic parameter view 2 ..... 534
4.1.10.9 Preview of PumpMon ..... 535
4.1.10.10 PumpMon limit value view ..... 537
4.1.10.11 PumpMon tolerances view ..... 538
4.1.10.12 Block icon for PumpMon ..... 539

4.2 PrDrpMon - Pressure drop monitoring ..... 540
4.2.1 Description of PrDrpMon ..... 540
4.2.2 PrDrpMon modes ..... 543
4.2.3 PrDrpMon mode of operation ..... 543
4.2.4 PrDrpMon functions ..... 546
4.2.5 PrDrpMon messaging ..... 550
4.2.6 PrDrpMon I/Os ..... 550
4.2.7 PrDrpMon block diagram ..... 555
4.2.8 Operator control and monitoring ..... 555
4.2.8.1 Views of PrDrpMon ..... 555
4.2.8.2 PrDrpMon standard view ..... 556
4.2.8.3 PrDrpMon limit view ..... 558
4.2.8.4 PrDrpMon parameter view ..... 559
4.2.8.5 Block icon for PrDrpMon ..... 560

4.3 Valve monitoring - VlvMon ..... 561
4.3.1 Description of VlvMon ..... 561
4.3.2 VlvMon modes ..... 565
4.3.3 VlvMon operating principle ..... 566

4.3.4 VlvMon functions ... 574
4.3.5 VlvMon error handling ... 581
4.3.6 VlvMon messaging ... 582
4.3.7 I/Os of VlvMon ... 582
4.3.8 VlvMon block diagram ... 596
4.3.9 VlvMon additional information ... 596
4.3.10 Operator control and monitoring ... 597
4.3.10.1 VlvMon views ... 597
4.3.10.2 VlvMon standard view ... 598
4.3.10.3 VlvMon flow characteristic curve view ... 599
4.3.10.4 VlvMon reaction time characteristic curve view ... 601
4.3.10.5 VlvMon histogram view ... 604
4.3.10.6 VlvMon limit view ... 606
4.3.10.7 VlvMon parameter 1 view ... 608
4.3.10.8 VlvMon parameter 2 view ... 610
4.3.10.9 Block icon for VlvMon ... 612

4.4 Stationary processes - SteadyState ... 614
4.4.1 Description of SteadyState ... 614
4.4.2 SteadyState modes ... 615
4.4.3 SteadyState operating principle ... 615
4.4.4 SteadyState functions ... 617
4.4.5 SteadyState error handling ... 620
4.4.6 SteadyState I/Os ... 620
4.4.7 SteadyState block diagram ... 622
4.4.8 Operator control and monitoring ... 623
4.4.8.1 Views of SteadyState ... 623
4.4.8.2 SteadyState standard view ... 623
4.4.8.3 SteadyState parameter view ... 624
4.4.8.4 SteadyState trend view ... 625
4.4.8.5 Block icon for SteadyState ... 625

4.5 Partial Stroke Test (PST) ... 627
4.5.1 Introduction ... 627
4.5.1.1 What is the Partial Stroke Test? ... 627
4.5.1.2 The Partial Stroke Test application ... 630
4.5.1.3 Hardware overview ... 632
4.5.1.4 Overview of engineering template for Partial Stroke Test ... 638
4.5.1.5 Overview of Partial Stroke Test user interface ... 640
4.5.2 Usage guidelines ... 641
4.5.2.1 Basic rules ... 641
4.5.2.2 Safety instructions ... 641
4.5.3 Getting Started ... 642
4.5.3.1 Hardware requirements ... 642
4.5.3.2 Software prerequisites ... 643
4.5.4 Configuration of the engineering template ... 643
4.5.4.1 Overview of Partial Stroke Test logic ... 643
4.5.4.2 Logic for the solenoid valve test (optional) ... 646
4.5.4.3 Adding the Partial Stroke Test engineering template to a STEP 7 project ... 648
4.5.4.4 Configure blocks ... 651
4.5.4.5 Compile and load logic of the Partial Stroke Test ... 652
4.5.5 Working with the engineering template ... 654
4.5.5.1 Working with the engineering template ... 654

4.5.5.2 Display Partial Stroke Test data ... 654
4.5.5.3 Changing the values of Partial Stroke Test parameters ... 655
4.5.5.4 Changing the values of solenoid valve test parameters ... 656
4.5.5.5 Starting or aborting the Partial Stroke Test manually ... 656
4.5.6 F_PST ... 656
4.5.7 F_PST_S ... 657
4.5.8 PST_Calc block ... 657
4.5.8.1 Description of PST_Calc ... 657
4.5.8.2 I/Os of PST_Calc ... 661
4.5.9 PST_OrQC block ... 662
4.5.9.1 Description of PST_OrQC ... 662
4.5.9.2 I/Os of PST_OrQC ... 663
4.5.10 PST_Time block ... 664
4.5.10.1 Description of PST_Time ... 664
4.5.10.2 I/Os of PST_Time ... 664
4.5.11 PST Block ... 665
4.5.11.1 Description of PST ... 665
4.5.11.2 PST operating modes ... 669
4.5.11.3 PST functions ... 669
4.5.11.4 PST error handling ... 671
4.5.11.5 PST reporting ... 672
4.5.11.6 PST I/Os ... 674
4.5.11.7 Operator control and monitoring ... 686
4.5.12 Create reports ... 692

**5 Monitoring blocks ... 701**

5.1 Comparison of large & small blocks ... 701
5.1.1 MonAnL compared to MonAnS ... 701
5.1.2 MonDiL compared to MonDiS ... 703

5.2 AV - Displaying and monitoring additional value ... 705
5.2.1 Description of AV ... 705
5.2.2 AV modes ... 707
5.2.3 AV functions ... 707
5.2.4 AV error handling ... 709
5.2.5 AV messaging ... 710
5.2.6 AV I/Os ... 711
5.2.7 AV block diagram ... 714

5.3 MonAnL - Monitoring of an analog process tag (Large) ... 715
5.3.1 Description of MonAnL ... 715
5.3.2 MonAnL modes ... 718
5.3.3 MonAnL functions ... 719
5.3.4 MonAnL error handling ... 726
5.3.5 MonAnL messaging ... 727
5.3.6 MonAnL I/Os ... 729
5.3.7 MonAnL block diagram ... 738
5.3.8 Operator control and monitoring ... 739
5.3.8.1 MonAnL views ... 739
5.3.8.2 MonAnL standard view ... 740
5.3.8.3 MonAnL limit view ... 743
5.3.8.4 MonAnL parameter view ... 745
5.3.8.5 MonAnL preview ... 746

5.3.8.6 MonAnL AddOn view ... 747
5.3.8.7 Block icon for MonAnL ... 748

5.4 MonAnS - Monitoring of an analog process tag (Small) ... 750
5.4.1 Description of MonAnS ... 750
5.4.2 MonAnS operating modes ... 752
5.4.3 MonAnS functions ... 753
5.4.4 MonAnS error handling ... 755
5.4.5 MonAnS messaging ... 756
5.4.6 MonAnS I/Os ... 758
5.4.7 MonAnS block diagram ... 762
5.4.8 Operator control and monitoring ... 762
5.4.8.1 MonAnS views ... 762
5.4.8.2 MonAnS standard view ... 763
5.4.8.3 MonAnS limit view ... 765
5.4.8.4 MonAnS parameter view ... 766
5.4.8.5 MonAnS preview ... 767
5.4.8.6 Block icon for MonAnS ... 768

5.5 MonDiL - Monitoring of a digital process tag (Large) ... 770
5.5.1 Description of MonDiL ... 770
5.5.2 MonDiL modes ... 773
5.5.3 MonDiL functions ... 774
5.5.4 MonDiL error handling ... 779
5.5.5 MonDiL messaging ... 780
5.5.6 MonDiL I/Os ... 781
5.5.7 MonDiL block diagram ... 786
5.5.8 Operator control and monitoring ... 787
5.5.8.1 MonDiL views ... 787
5.5.8.2 MonDiL standard view ... 788
5.5.8.3 MonDiL parameter view ... 790
5.5.8.4 MonDiL preview ... 792
5.5.8.5 Block icon for MonDiL ... 793

5.6 MonDiS - Monitoring of a digital process tag (Small) ... 796
5.6.1 Description of MonDiS ... 796
5.6.2 MonDiS operating modes ... 799
5.6.3 MonDiS functions ... 800
5.6.4 MonDiS error handling ... 803
5.6.5 MonDiS messaging ... 804
5.6.6 MonDiS I/Os ... 805
5.6.7 MonDiS block diagram ... 808
5.6.8 Operator control and monitoring ... 809
5.6.8.1 MonDiS views ... 809
5.6.8.2 MonDiS standard view ... 809
5.6.8.3 MonDiS parameter view ... 811
5.6.8.4 MonDiS preview ... 812
5.6.8.5 Block icon for MonDiS ... 813

5.7 MonDi08 - Monitoring 8 digital process tags ... 816
5.7.1 Description of MonDi08 ... 816
5.7.2 MonDi08 modes ... 818
5.7.3 MonDi08 functions ... 819
5.7.4 MonDi08 error handling ... 822

5.7.5 MonDi08 messaging ... 823
5.7.6 MonDi08 I/Os ... 824
5.7.7 MonDi08 block diagram ... 829
5.7.8 Operator control and monitoring ... 829
5.7.8.1 MonDi08 views ... 829
5.7.8.2 MonDi08 standard view ... 830
5.7.8.3 MonDi08 parameter view ... 832
5.7.8.4 MonDi08 preview ... 833
5.7.8.5 Block icon for MonDi08 ... 834

**6 Controller blocks ... 837**

6.1 Comparison of large & small blocks ... 837
6.1.1 PIDConL compared to PIDConS ... 837

6.2 ConPerMon - Monitoring of the control performance of control loops ... 841
6.2.1 Description of ConPerMon ... 841
6.2.2 ConPerMon modes ... 845
6.2.3 ConPerMon functions ... 845
6.2.4 ConPerMon error handling ... 857
6.2.5 ConPerMon messaging ... 857
6.2.6 ConPerMon I/Os ... 860
6.2.7 ConPerMon block diagram ... 866
6.2.8 Operator control and monitoring ... 866
6.2.8.1 ConPerMon views ... 866
6.2.8.2 ConPerMon standard view ... 867
6.2.8.3 ConPerMon limit view ... 869
6.2.8.4 ConPerMon parameter view ... 870
6.2.8.5 ConPerMon preview ... 871
6.2.8.6 ConPerMon setpoint view ... 872
6.2.8.7 Block icon for ConPerMon ... 874

6.3 FmCont - Interface to module FM 355 ... 876
6.3.1 Description of FmCont ... 876
6.3.2 FmCont modes ... 880
6.3.3 FmCont functions ... 882
6.3.4 FmCont error handling ... 892
6.3.5 FmCont messaging ... 894
6.3.6 FmCont I/Os ... 897
6.3.7 FmCont block diagram ... 913
6.3.8 Operator control and monitoring ... 915
6.3.8.1 FmCont views ... 915

6.4 FmTemp - Interface to temperature controller modules FM 355-2 ... 916
6.4.1 Description of FmTemp ... 916
6.4.2 FmTemp modes ... 920
6.4.3 FmTemp functions ... 922
6.4.4 FmTemp error handling ... 934
6.4.5 FmTemp messaging ... 935
6.4.6 FmTemp I/Os ... 938
6.4.7 FmTemp block diagram ... 956
6.4.8 Operator control and monitoring ... 958
6.4.8.1 FmTemp views ... 958

6.5 GainSched - Adapting parameter values for a PID controller ... 959

6.5.1 Description of GainSched ... 959
6.5.2 GainSched modes ... 961
6.5.3 GainSched functions ... 962
6.5.4 GainSched error handling ... 963
6.5.5 GainSched messaging ... 963
6.5.6 GainSched I/Os ... 964
6.5.7 GainSched block diagram ... 966
6.5.8 Operator control and monitoring ... 967
6.5.8.1 GainSched views ... 967
6.5.8.2 GainSched standard view ... 967
6.5.8.3 GainSched parameter view ... 968
6.5.8.4 GainSched preview ... 969

6.6 ModPreCon - Model predictive controller ... 970
6.6.1 Description of ModPreCon ... 970
6.6.2 ModPreCon modes ... 975
6.6.3 ModPreCon functions ... 976
6.6.4 ModPreCon error handling ... 989
6.6.5 ModPreCon messaging ... 990
6.6.6 ModPreCon I/Os ... 990
6.6.7 ModPreCon block diagram ... 1001
6.6.8 Operator control and monitoring ... 1002
6.6.8.1 ModPreCon views ... 1002
6.6.8.2 ModPreCon standard view ... 1003
6.6.8.3 ModPreCon parameter view ... 1008
6.6.8.4 ModPreCon parameter view channel 1 to 4 ... 1010
6.6.8.5 ModPreCon preview ... 1011
6.6.8.6 ModPreCon trend view ... 1013
6.6.8.7 Block icon for ModPreCon ... 1014

6.7 PIDConL - Continuous PID controller (Large) ... 1017
6.7.1 Description of PIDConL ... 1017
6.7.2 PIDConL modes ... 1022
6.7.3 PIDConL functions ... 1024
6.7.4 PIDConL error handling ... 1034
6.7.5 PIDConL messaging ... 1036
6.7.6 PIDConL I/Os ... 1039
6.7.7 PIDConL block diagram ... 1058
6.7.8 Operator control and monitoring ... 1060
6.7.8.1 PIDConL views ... 1060
6.7.8.2 PIDConL, PIDConS and PIDConR standard views ... 1061
6.7.8.3 PIDConL, PIDConS and PIDConR previews ... 1066

6.8 PIDConS - Continuous PID controller (Small) ... 1069
6.8.1 Description of PIDConS ... 1069
6.8.2 PIDConS modes ... 1072
6.8.3 PIDConS functions ... 1073
6.8.4 PIDConS error handling ... 1079
6.8.5 PIDConS messaging ... 1080
6.8.6 PIDConS I/Os ... 1082
6.8.7 PIDConS block diagram ... 1091
6.8.8 Operator control and monitoring ... 1093
6.8.8.1 PIDConS views ... 1093

6.9 PIDConR - Continuous PID controller with external reset ... 1094
6.9.1 Description of PIDConR ... 1094
6.9.2 PIDConR modes ... 1100
6.9.3 PIDConR functions ... 1103
6.9.4 PIDConR error handling ... 1117
6.9.5 PIDConR messaging ... 1118
6.9.6 PIDConR I/Os ... 1121
6.9.7 PIDConR block diagram ... 1140
6.9.8 Operator control and monitoring ... 1140
6.9.8.1 PIDConR views ... 1140

6.10 PIDStepL - Step controller ... 1141
6.10.1 Description of PIDStepL ... 1141
6.10.2 PIDStepL modes ... 1146
6.10.3 PIDStepL functions ... 1147
6.10.4 PIDStepL error handling ... 1159
6.10.5 PIDStepL messaging ... 1160
6.10.6 PIDStepL I/Os ... 1163
6.10.7 PIDStepL block diagram ... 1179
6.10.8 Operator control and monitoring ... 1184
6.10.8.1 PIDStepL views ... 1184
6.10.8.2 PIDStepL standard view without position feedback ... 1185
6.10.8.3 PIDStepL standard view with position feedback ... 1189
6.10.8.4 PIDStepL preview ... 1193

6.11 Ratio - Ratio controlling ... 1196
6.11.1 Description of Ratio ... 1196
6.11.2 Ratio modes ... 1198
6.11.3 Ratio functions ... 1198
6.11.4 Ratio error handling ... 1201
6.11.5 Ratio messaging ... 1202
6.11.6 Ratio I/Os ... 1202
6.11.7 Ratio block diagram ... 1207
6.11.8 Operator control and monitoring ... 1207
6.11.8.1 Ratio views ... 1207
6.11.8.2 Ratio standard view ... 1208
6.11.8.3 Ratio parameter view ... 1211
6.11.8.4 Ratio preview ... 1212
6.11.8.5 Block icon for Ratio ... 1213

6.12 SplRange - Signal splitter ... 1216
6.12.1 Description of SplRange ... 1216
6.12.2 SplRange modes ... 1217
6.12.3 SplRange functions ... 1218
6.12.4 SplRange error handling ... 1221
6.12.5 SplRange messaging ... 1221
6.12.6 SplRange I/Os ... 1222
6.12.7 SplRange block diagram ... 1223

6.13 AutoExcitation - Process trigger for predictive controller ... 1224
6.13.1 Description of AutoExcitation ... 1224
6.13.2 AutoExcitation modes ... 1226
6.13.3 AutoExcitation functions ... 1226
6.13.4 AutoExcitation error handling ... 1227

6.13.5 AutoExcitation messaging ... 1227
6.13.6 AutoExcitation I/Os ... 1227
6.13.7 AutoExcitation block diagram ... 1228

6.14 LPOptim - Optimization after traversing the linear programming ... 1229
6.14.1 Description of LPOptim ... 1229
6.14.2 LPOptim modes ... 1230
6.14.3 LPOptim functions ... 1230
6.14.4 LPOptim error handling ... 1230
6.14.5 LPOptim messaging ... 1230
6.14.6 LPOptim I/Os ... 1230
6.14.7 LPOptim block diagram ... 1233

6.15 MPC10x10 - Large predictive controller ... 1234
6.15.1 Description of MPC10x10 ... 1234
6.15.2 MPC10x10 modes ... 1241
6.15.3 MPC10x10 functions ... 1243
6.15.4 MPC10x10 error handling ... 1253
6.15.5 MPC10x10 messaging ... 1254
6.15.6 MPC10x10 I/Os ... 1254
6.15.7 MPC10x10 block diagram ... 1260
6.15.8 Operator control and monitoring ... 1261
6.15.8.1 MPC10x10 views ... 1261
6.15.8.2 MPC10x10 standard view ... 1261
6.15.8.3 MPC10x10 parameter view ... 1267
6.15.8.4 MPC10x10 CV parameter view ... 1269
6.15.8.5 MPC10x10 MV parameter view ... 1270
6.15.8.6 MPC10x10 preview ... 1271
6.15.8.7 MPC10x10 trend view ... 1273
6.15.8.8 Block icon for MPC10x10 ... 1274

6.16 KalFilt - State observer ... 1276
6.16.1 Description of KalFilt ... 1276
6.16.2 KalFilt modes ... 1282
6.16.3 KalFilt functions ... 1284
6.16.4 KalFilt error handling ... 1286
6.16.5 KalFilt I/Os ... 1287
6.16.6 KalFilt messaging ... 1292
6.16.7 KalFilt block diagram ... 1292
6.16.8 Operator control and monitoring ... 1293
6.16.8.1 KalFilt views ... 1293
6.16.8.2 KalFilt standard view ... 1293
6.16.8.3 KalFilt parameter view ... 1297
6.16.8.4 KalFilt parameter view 2 ... 1298
6.16.8.5 KalFilt preview ... 1299
6.16.8.6 KalFilt measurements view ... 1301
6.16.8.7 KalFilt trend view ... 1302
6.16.8.8 Block icon for KalFilt ... 1302

**7 Dosing blocks ... 1305**

7.1 DoseL - Dosing device ... 1305
7.1.1 Description of DoseL ... 1305
7.1.2 DoseL modes ... 1310
7.1.3 DoseL functions ... 1312

7.1.4 DoseL error handling ... 1327
7.1.5 DoseL messaging ... 1329
7.1.6 DoseL I/Os ... 1332
7.1.7 DoseL block diagram ... 1348
7.1.8 Operator control and monitoring ... 1349
7.1.8.1 DoseL views ... 1349
7.1.8.2 DoseL standard view ... 1350
7.1.8.3 DoseL limit view ... 1354
7.1.8.4 DoseL parameter view ... 1356
7.1.8.5 DoseL flow setpoint view ... 1358
7.1.8.6 DoseL quantity setpoint view ... 1361
7.1.8.7 DoseL preview ... 1363
7.1.8.8 Block icon for DoseL ... 1366

**8 Motor and valve blocks ... 1369**

8.1 Comparison of large & small blocks ... 1369
8.1.1 MotL compared to MotS ... 1369
8.1.2 VlvL compared to VlvS ... 1372
8.1.3 ShrdResL compared to ShrdResS ... 1375

8.2 MotL - Motor (Large) ... 1376
8.2.1 Description of MotL ... 1376
8.2.2 MotL modes ... 1380
8.2.3 MotL functions ... 1382
8.2.4 MotL error handling ... 1390
8.2.5 MotL messaging ... 1392
8.2.6 MotL I/Os ... 1394
8.2.7 MotL block diagram ... 1404
8.2.8 Operator control and monitoring ... 1404
8.2.8.1 MotL views ... 1404
8.2.8.2 MotL standard view ... 1405
8.2.8.3 MotL preview ... 1408
8.2.8.4 Block icon for MotL ... 1412

8.3 MotS - Motor (Small) ... 1415
8.3.1 Description of MotS ... 1415
8.3.2 MotS modes ... 1418
8.3.3 MotS functions ... 1419
8.3.4 MotS error handling ... 1424
8.3.5 MotS messaging ... 1425
8.3.6 MotS I/Os ... 1426
8.3.7 MotS block diagram ... 1432
8.3.8 Operator control and monitoring ... 1432
8.3.8.1 MotS views ... 1432
8.3.8.2 MotS standard view ... 1433
8.3.8.3 MotS preview ... 1436
8.3.8.4 Block icon for MotS ... 1438

8.4 MotRevL - Reversible motor ... 1440
8.4.1 Description of MotRevL ... 1440
8.4.2 MotRevL modes ... 1445
8.4.3 MotRevL functions ... 1447
8.4.4 MotRevL error handling ... 1457
8.4.5 MotRevL messaging ... 1461

8.4.6 MotRevL I/Os ... 1463
8.4.7 MotRevL block diagram ... 1474
8.4.8 Operator control and monitoring ... 1475
8.4.8.1 MotRevL views ... 1475
8.4.8.2 MotRevL standard view ... 1475
8.4.8.3 MotRevL preview ... 1479
8.4.8.4 Block icon for MotRevL ... 1485

8.5 MotSpdCL - Controllable reversible motor ... 1488
8.5.1 Description of MotSpdCL ... 1488
8.5.2 MotSpdCL modes ... 1494
8.5.3 MotSpdCL functions ... 1496
8.5.4 MotSpdCL error handling ... 1512
8.5.5 MotSpdCL messaging ... 1516
8.5.6 MotSpdCL I/Os ... 1518
8.5.7 MotSpdCL block diagram ... 1535
8.5.8 Operator control and monitoring ... 1535
8.5.8.1 MotSpdCL views ... 1535
8.5.8.2 MotSpdCL standard view ... 1536
8.5.8.3 MotSpdCL preview ... 1542
8.5.8.4 MotSpdCL limit view for readback values ... 1546
8.5.8.5 MotSpdCL parameter view ... 1548
8.5.8.6 MotSpdCL trend view ... 1550
8.5.8.7 MotSpdCL AddOn view ... 1552
8.5.8.8 Block icon for MotSpdCL ... 1552

8.6 MotSpdL - Two-speed motor ... 1556
8.6.1 Description of MotSpdL ... 1556
8.6.2 MotSpdL modes ... 1561
8.6.3 MotSpdL functions ... 1563
8.6.4 MotSpdL error handling ... 1574
8.6.5 MotSpdL messaging ... 1577
8.6.6 MotSpdL I/Os ... 1578
8.6.7 MotSpdL block diagram ... 1590
8.6.8 Operator control and monitoring ... 1590
8.6.8.1 MotSpdL views ... 1590
8.6.8.2 MotSpdL standard view ... 1591
8.6.8.3 MotSpdL preview ... 1595
8.6.8.4 Block icon for MotSpdL ... 1600

8.7 ShrdResL - Multiplexer for shared resources (Large) ... 1603
8.7.1 Description of ShrdResL ... 1603
8.7.2 ShrdResL modes ... 1606
8.7.3 ShrdResL functions ... 1607
8.7.4 ShrdResL error handling ... 1613
8.7.5 ShrdResL messaging ... 1613
8.7.6 ShrdResL I/Os ... 1614
8.7.7 ShrdResL block diagram ... 1619
8.7.8 Operator control and monitoring ... 1619
8.7.8.1 ShrdResL views ... 1619
8.7.8.2 ShrdResL standard view ... 1620
8.7.8.3 ShrdResL general preview ... 1622
8.7.8.4 ShrdResL preview ... 1622
8.7.8.5 ShrdResL parameter view ... 1624

8.7.8.6 Block icon for ShrdResL ... 1625

8.8 ShrdResS - Multiplexer for shared resources (Small) ... 1626
8.8.1 Description for ShrdResS ... 1626
8.8.2 ShrdResS modes ... 1629
8.8.3 ShrdResS functions ... 1629
8.8.4 ShrdResS error handling ... 1632
8.8.5 ShrdResS messaging ... 1632
8.8.6 ShrdResS I/Os ... 1633
8.8.7 ShrdResS block diagram ... 1641
8.8.8 Operator control and monitoring ... 1642
8.8.8.1 ShrdResS views ... 1642
8.8.8.2 ShrdResS standard view ... 1642
8.8.8.3 ShrdResS preview ... 1644
8.8.8.4 Block icon for ShrdResS ... 1645

8.9 Vlv2WayL - Two-way valve ... 1647
8.9.1 Description of Vlv2WayL ... 1647
8.9.2 Vlv2WayL modes ... 1651
8.9.3 Vlv2WayL functions ... 1653
8.9.4 Vlv2WayL error handling ... 1660
8.9.5 Vlv2WayL messaging ... 1663
8.9.6 Vlv2WayL I/Os ... 1664
8.9.7 Vlv2WayL block diagram ... 1676
8.9.8 Operator control and monitoring ... 1677
8.9.8.1 Vlv2WayL views ... 1677
8.9.8.2 Vlv2WayL standard view ... 1677
8.9.8.3 Vlv2WayL parameter view ... 1681
8.9.8.4 Vlv2WayL preview ... 1683
8.9.8.5 Block icon for Vlv2WayL ... 1688

8.10 VlvL - Valve (Large) ... 1691
8.10.1 Description of VlvL ... 1691
8.10.2 VlvL modes ... 1694
8.10.3 VlvL functions ... 1696
8.10.4 VlvL error handling ... 1703
8.10.5 VlvL messaging ... 1705
8.10.6 VlvL I/Os ... 1707
8.10.7 VlvL block diagram ... 1715
8.10.8 Operator control and monitoring ... 1716
8.10.8.1 VlvL views ... 1716
8.10.8.2 VlvL standard view ... 1717
8.10.8.3 VlvL preview ... 1721
8.10.8.4 VlvL block icon ... 1725

8.11 VlvS - Valve (small) ... 1728
8.11.1 Description of VlvS ... 1728
8.11.2 VlvS modes ... 1731
8.11.3 VlvS functions ... 1732
8.11.4 VlvS error handling ... 1736
8.11.5 VlvS reporting ... 1738
8.11.6 VlvS I/Os ... 1739
8.11.7 VlvS block diagram ... 1745
8.11.8 Operator control and monitoring ... 1746

8.11.8.1 VlvS views ... 1746
8.11.8.2 VlvS standard view ... 1746
8.11.8.3 VlvS preview ... 1749
8.11.8.4 VlvS block icon ... 1751

8.12 VlvMotL - Motor valve ... 1754
8.12.1 Description of VlvMotL ... 1754
8.12.2 VlvMotL modes ... 1759
8.12.3 VlvMotL functions ... 1762
8.12.4 VlvMotL error handling ... 1774
8.12.5 VlvMotL messaging ... 1777
8.12.6 VlvMotL I/Os ... 1778
8.12.7 VlvMotL block diagram ... 1791
8.12.8 Operator control and monitoring ... 1791
8.12.8.1 VlvMotL views ... 1791
8.12.8.2 VlvMotL standard view ... 1792
8.12.8.3 VlvMotL parameter view ... 1797
8.12.8.4 VlvMotL preview ... 1799
8.12.8.5 Block icon for VlvMotL ... 1805

8.13 VlvAnL - Control valve ... 1808
8.13.1 Description of VlvAnL ... 1808
8.13.2 VlvAnL modes ... 1813
8.13.3 VlvAnL functions ... 1816
8.13.4 VlvAnL error handling ... 1832
8.13.5 VlvAnL messaging ... 1833
8.13.6 VlvAnL I/Os ... 1835
8.13.7 VlvAnL block diagram ... 1850
8.13.8 Operator control and monitoring ... 1851
8.13.8.1 VlvAnL views ... 1851
8.13.8.2 VlvAnL standard view with auxiliary valve ... 1852
8.13.8.3 VlvAnL standard view without auxiliary valve ... 1857
8.13.8.4 VlvAnL limit view ... 1863
8.13.8.5 VlvAnL preview ... 1865
8.13.8.6 VlvAnL parameter view ... 1871
8.13.8.7 Block icon for VlvAnL ... 1872

8.14 VlvPosL - Valve positioner ... 1876
8.14.1 Description of VlvPosL ... 1876
8.14.2 VlvPosL modes ... 1882
8.14.3 VlvPosL functions ... 1884
8.14.4 VlvPosL error handling ... 1908
8.14.5 VlvPosL messaging ... 1910
8.14.6 VlvPosL I/Os ... 1912
8.14.7 VlvPosL block diagram ... 1928
8.14.8 Operator control and monitoring ... 1929
8.14.8.1 VlvPosL views ... 1929
8.14.8.2 VlvPosL standard view ... 1930
8.14.8.3 VlvPosL limit value view ... 1935
8.14.8.4 VlvPosL parameter view ... 1937
8.14.8.5 VlvPosL preview ... 1941
8.14.8.6 Block icon for VlvPosL ... 1947

**9 Interlock blocks ..... 1949**

9.1 Intlk02 - Interlock display with 2 input signals ..... 1949
9.1.1 Description of Intlk02 ..... 1949
9.1.2 Intlk02 modes ..... 1952
9.1.3 Intlk02 functions ..... 1952
9.1.4 Intlk02 error handling ..... 1956
9.1.5 Intlk02 messaging ..... 1957
9.1.6 Intlk02 I/Os ..... 1957
9.1.7 Intlk02 block diagram ..... 1961
9.1.8 Operator control and monitoring ..... 1961
9.1.8.1 Interlock block views ..... 1961

9.2 Intlk04 - Interlock display with 4 input signals ..... 1962
9.2.1 Description of Intlk04 ..... 1962
9.2.2 Intlk04 modes ..... 1965
9.2.3 Intlk04 functions ..... 1965
9.2.4 Intlk04 error handling ..... 1969
9.2.5 Intlk04 messaging ..... 1970
9.2.6 Intlk04 I/Os ..... 1970
9.2.7 Intlk04 block diagram ..... 1974
9.2.8 Operator control and monitoring ..... 1975
9.2.8.1 Interlock block views ..... 1975

9.3 Intlk08 - Interlock display with 8 input signals ..... 1976
9.3.1 Description of Intlk08 ..... 1976
9.3.2 Intlk08 modes ..... 1979
9.3.3 Intlk08 functions ..... 1980
9.3.4 Intlk08 error handling ..... 1984
9.3.5 Intlk08 messaging ..... 1985
9.3.6 Intlk08 I/Os ..... 1985
9.3.7 Intlk08 block diagram ..... 1991
9.3.8 Operator control and monitoring ..... 1991
9.3.8.1 Interlock block views ..... 1991

9.4 Intlk16 - Interlock display with 16 input signals ..... 1992
9.4.1 Description of Intlk16 ..... 1992
9.4.2 Intlk16 modes ..... 1997
9.4.3 Intlk16 functions ..... 1997
9.4.4 Intlk16 error handling ..... 2002
9.4.5 Intlk16 messaging ..... 2002
9.4.6 Intlk16 I/Os ..... 2003
9.4.7 Intlk16 block diagram ..... 2010
9.4.8 Operator control and monitoring ..... 2011
9.4.8.1 Interlock block views ..... 2011

9.5 FirstIn - Transform output Intlck FirstIn for message associated value ..... 2012
9.5.1 Description of FirstIn ..... 2012
9.5.2 FirstIn modes ..... 2013
9.5.3 FirstIn functions ..... 2014
9.5.4 FirstIn error handling ..... 2014
9.5.5 FirstIn messaging ..... 2015
9.5.6 FirstIn I/Os ..... 2015
9.5.7 FirstIn block diagram ..... 2016

**10 Message blocks ... 2017**

10.1 Event - Creating messages ... 2017
10.1.1 Description of Event ... 2017
10.1.2 Event modes ... 2020
10.1.3 Event functions ... 2020
10.1.4 Event error handling ... 2022
10.1.5 Event messaging ... 2023
10.1.6 Event I/Os ... 2025
10.1.7 Event block diagram ... 2029

10.2 EventNck - Generating messages without acknowledgment ... 2030
10.2.1 Description of EventNck ... 2030
10.2.2 EventNck modes ... 2032
10.2.3 EventNck functions ... 2033
10.2.4 EventNck error handling ... 2035
10.2.5 EventNck messaging ... 2036
10.2.6 EventNck I/Os ... 2037
10.2.7 EventNck block diagram ... 2041

10.3 EventTs - Creating messages with time stamp ... 2042
10.3.1 Description of EventTs ... 2042
10.3.2 EventTs modes ... 2045
10.3.3 EventTs functions ... 2046
10.3.4 EventTs error handling ... 2048
10.3.5 EventTs messaging ... 2049
10.3.6 EventTs I/Os ... 2052
10.3.7 EventTs block diagram ... 2056

10.4 Event16Ts - Creating 16 messages with time stamp ... 2057
10.4.1 Description of Event16Ts ... 2057
10.4.2 Event16Ts operating modes ... 2062
10.4.3 Event16Ts functions ... 2062
10.4.4 Event16Ts error handling ... 2065
10.4.5 Event16Ts messages ... 2065
10.4.6 Event16Ts I/Os ... 2071
10.4.7 Event16Ts block diagram ... 2073

**11 Counter blocks ... 2075**

11.1 CountScL - Counter with up and down counting direction ... 2075
11.1.1 Description of CountScL ... 2075
11.1.2 CountScL modes ... 2079
11.1.3 CountScL functions ... 2080
11.1.4 CountScL error handling ... 2084
11.1.5 CountScL messaging ... 2084
11.1.6 CountScL I/Os ... 2085
11.1.7 CountScL block diagram ... 2090
11.1.8 Operator control and monitoring ... 2090
11.1.8.1 CountScL views ... 2090
11.1.8.2 CountScL standard view ... 2091
11.1.8.3 CountScL limit view ... 2093
11.1.8.4 CountScL parameter view ... 2094
11.1.8.5 CountScL preview ... 2095
11.1.8.6 Block icon for CountScL ... 2096

11.2 CountOh - Determining runtime ... 2098
11.2.1 Description of CountOh ... 2098
11.2.2 CountOh modes ... 2102
11.2.3 CountOh functions ... 2103
11.2.4 CountOh error handling ... 2107
11.2.5 CountOh messaging ... 2108
11.2.6 CountOh I/Os ... 2109
11.2.7 CountOh block diagram ... 2114
11.2.8 Operator control and monitoring ... 2115
11.2.8.1 CountOh views ... 2115
11.2.8.2 CountOh standard view ... 2116
11.2.8.3 CountOh limit view ... 2118
11.2.8.4 CountOh parameter view ... 2119
11.2.8.5 CountOh preview ... 2120
11.2.8.6 Block icon for CountOh ... 2121

11.3 TotalL - Additive counter with upward or downward counting direction (totalizer) ... 2123
11.3.1 Description of TotalL ... 2123
11.3.2 TotalL operating modes ... 2132
11.3.3 TotalL functions ... 2133
11.3.4 TotalL error handling ... 2138
11.3.5 TotalL messaging ... 2139
11.3.6 TotalL I/Os ... 2140
11.3.7 TotalL block diagram ... 2146
11.3.8 Operator control and monitoring ... 2146
11.3.8.1 TotalL views ... 2146
11.3.8.2 TotalL standard view ... 2147
11.3.8.3 TotalL limit view ... 2150
11.3.8.4 TotalL parameter view ... 2151
11.3.8.5 TotalL preview ... 2153
11.3.8.6 TotalL block icon ... 2154

11.4 CntOhSc - Runtime determination and counters with counting direction "up" ... 2156
11.4.1 Description of CntOhSc ... 2156
11.4.2 CntOhSc operating modes ... 2158
11.4.3 CntOhSc functions ... 2158
11.4.4 CntOhSc error handling ... 2161
11.4.5 CntOhSc messaging ... 2161
11.4.6 CntOhSc I/Os ... 2161
11.4.7 CntOhSc block diagram ... 2165
11.4.8 Operator control and monitoring ... 2165
11.4.8.1 CntOhSc views ... 2165
11.4.8.2 CntOhSc standard view ... 2166
11.4.8.3 CntOhSc limit view ... 2168
11.4.8.4 CntOhSc preview ... 2169
11.4.8.5 Block icon for CntOhSc ... 2170

**12 Timers ... 2173**

12.1 TimerP - Time delays signal forwarding / pulse generator ... 2173
12.1.1 Description of TimerP ... 2173
12.1.2 TimerP modes ... 2174
12.1.3 TimerP functions ... 2174
12.1.4 TimerP error handling ... 2177

12.1.5 TimerP messaging ... 2178
12.1.6 TimerP I/Os ... 2178
12.1.7 TimerP block diagram ... 2180

12.2 TimeTrig - Calculations with the date formats DT and TIME ... 2181
12.2.1 Description of TimeTrig ... 2181
12.2.2 TimeTrig modes ... 2183
12.2.3 TimeTrig functions ... 2184
12.2.4 TimeTrig error handling ... 2191
12.2.5 TimeTrig messaging ... 2192
12.2.6 TimeTrig I/Os ... 2192
12.2.7 TimeTrig block diagram ... 2197
12.2.8 Operator control and monitoring ... 2197
12.2.8.1 TimeTrig views ... 2197
12.2.8.2 TimeTrig standard view ... 2198
12.2.8.3 TimeTrig parameter view ... 2201
12.2.8.4 TimeTrig preview ... 2203
12.2.8.5 Block icon for TimeTrig ... 2205

**13 Mathematical blocks ... 2207**

13.1 AbsR - Absolute value of a real value ... 2207
13.1.1 Description of AbsR ... 2207
13.1.2 AbsR modes ... 2208
13.1.3 AbsR functions ... 2208
13.1.4 AbsR error handling ... 2208
13.1.5 AbsR messaging ... 2208
13.1.6 AbsR I/Os ... 2208
13.1.7 AbsR block diagram ... 2209

13.2 Add04 - Adder with 4 values ... 2210
13.2.1 Description of Add04 ... 2210
13.2.2 Add04 modes ... 2211
13.2.3 Add04 functions ... 2211
13.2.4 Add04 error handling ... 2212
13.2.5 Add04 messaging ... 2212
13.2.6 Add04 I/Os ... 2213
13.2.7 Add04 block diagram ... 2214

13.3 Add08 - Adder with 8 values ... 2215
13.3.1 Description of Add08 ... 2215
13.3.2 Add08 modes ... 2216
13.3.3 Add08 functions ... 2216
13.3.4 Add08 error handling ... 2217
13.3.5 Add08 messaging ... 2217
13.3.6 Add08 I/Os ... 2218
13.3.7 Add08 block diagram ... 2219

13.4 Average - Mean value calculation ... 2220
13.4.1 Description of Average ... 2220
13.4.2 Average modes ... 2221
13.4.3 Average functions ... 2222
13.4.4 Average error handling ... 2222
13.4.5 Average messaging ... 2223
13.4.6 Average I/Os ... 2224

13.4.7 Average block diagram ... 2225

13.5 DeadTime - Delayed signal output ... 2226
13.5.1 Description of DeadTime ... 2226
13.5.2 DeadTime modes ... 2228
13.5.3 DeadTime functions ... 2228
13.5.4 DeadTime error handling ... 2229
13.5.5 DeadTime messaging ... 2230
13.5.6 DeadTime I/Os ... 2230
13.5.7 DeadTime block diagram ... 2231

13.6 Derivative - Obtaining a derivative ... 2233
13.6.1 Description of Derivative ... 2233
13.6.2 Derivative modes ... 2235
13.6.3 Derivative functions ... 2236
13.6.4 Derivative error handling ... 2237
13.6.5 Derivative messaging ... 2237
13.6.6 Derivative I/Os ... 2238
13.6.7 Derivative block diagram ... 2239

13.7 Div02 - Division of two values ... 2240
13.7.1 Description of Div02 ... 2240
13.7.2 Div02 modes ... 2241
13.7.3 Div02 functions ... 2241
13.7.4 Div02 error handling ... 2242
13.7.5 Div02 messaging ... 2243
13.7.6 Div02 I/Os ... 2243
13.7.7 Div02 block diagram ... 2244

13.8 FlowCorr - Flow correction ... 2245
13.8.1 Description of FlowCorr ... 2245
13.8.2 FlowCorr modes ... 2247
13.8.3 FlowCorr functions ... 2247
13.8.4 FlowCorr error handling ... 2249
13.8.5 FlowCorr messaging ... 2250
13.8.6 FlowCorr I/Os ... 2250
13.8.7 FlowCorr block diagram ... 2252

13.9 Integral - Generating a time integral ... 2253
13.9.1 Description of Integral ... 2253
13.9.2 Integral modes ... 2254
13.9.3 Integral functions ... 2255
13.9.4 Integral error handling ... 2257
13.9.5 Integral messaging ... 2258
13.9.6 Integral I/Os ... 2258
13.9.7 Integral block diagram ... 2259

13.10 Lag - Low-pass filter ... 2261
13.10.1 Description of Lag ... 2261
13.10.2 Lag modes ... 2263
13.10.3 Lag functions ... 2263
13.10.4 Lag error handling ... 2264
13.10.5 Lag messaging ... 2265
13.10.6 Lag I/Os ... 2266
13.10.7 Lag block diagram ... 2267

13.11 MeanTime - Averaging ... 2268
13.11.1 Description of MeanTime ... 2268
13.11.2 MeanTime modes ... 2269
13.11.3 MeanTime functions ... 2270
13.11.4 MeanTime error handling ... 2272
13.11.5 MeanTime messaging ... 2272
13.11.6 MeanTime I/Os ... 2273
13.11.7 MeanTime block diagram ... 2274

13.12 Mul04 - Multiplier with 4 values ... 2275
13.12.1 Description of Mul04 ... 2275
13.12.2 Mul04 modes ... 2276
13.12.3 Mul04 functions ... 2277
13.12.4 Mul04 error handling ... 2278
13.12.5 Mul04 messaging ... 2278
13.12.6 Mul04 I/Os ... 2278
13.12.7 Mul04 block diagram ... 2279

13.13 Mul08 - Multiplier with 8 values ... 2281
13.13.1 Description of Mul08 ... 2281
13.13.2 Mul08 modes ... 2282
13.13.3 Mul08 functions ... 2282
13.13.4 Mul08 error handling ... 2283
13.13.5 Mul08 messaging ... 2284
13.13.6 Mul08 I/Os ... 2284
13.13.7 Mul08 block diagram ... 2285

13.14 Polygon - Converting the first signal (non-linear) ... 2287
13.14.1 Description of Polygon ... 2287
13.14.2 Polygon modes ... 2290
13.14.3 Polygon functions ... 2290
13.14.4 Polygon error handling ... 2291
13.14.5 Polygon messaging ... 2292
13.14.6 Polygon I/Os ... 2293
13.14.7 Polygon block diagram ... 2296

13.15 Smooth - Low pass filter ... 2297
13.15.1 Description of Smooth ... 2297
13.15.2 Smooth modes ... 2298
13.15.3 Smooth functions ... 2298
13.15.4 Smooth error handling ... 2299
13.15.5 Smooth messaging ... 2300
13.15.6 Smooth I/Os ... 2301
13.15.7 Smooth block diagram ... 2302

13.16 SqrRoot - Derive the root of a value ... 2303
13.16.1 Description of SqrRoot ... 2303
13.16.2 SqrRoot modes ... 2304
13.16.3 SqrRoot functions ... 2304
13.16.4 SqrRoot error handling ... 2305
13.16.5 SqrRoot messaging ... 2305
13.16.6 SqrRoot I/Os ... 2306
13.16.7 SqrRoot block diagram ... 2307

13.17 Sub02 - Subtracting two values ... 2308

13.17.1 Description of Sub02 ... 2308
13.17.2 Sub02 modes ... 2309
13.17.3 Sub02 functions ... 2309
13.17.4 Sub02 error handling ... 2310
13.17.5 Sub02 messaging ... 2310
13.17.6 Sub02 I/Os ... 2311
13.17.7 Sub02 block diagram ... 2312

13.18 PowerXY - General power function ... 2313
13.18.1 Description of PowerXY ... 2313
13.18.2 PowerXY modes ... 2314
13.18.3 PowerXY functions ... 2315
13.18.4 PowerXY error handling ... 2315
13.18.5 PowerXY messaging ... 2316
13.18.6 PowerXY I/Os ... 2316
13.18.7 PowerXY block diagrams ... 2317

**14 Analog logic blocks ... 2319**

14.1 CompAn02 - Comparison of two analog values ... 2319
14.1.1 Description of CompAn02 ... 2319
14.1.2 CompAn02 modes ... 2320
14.1.3 CompAn02 functions ... 2320
14.1.4 CompAn02 error handling ... 2321
14.1.5 CompAn02 messaging ... 2322
14.1.6 CompAn02 I/Os ... 2322
14.1.7 CompAn02 block diagram ... 2323

14.2 Limit - Limiting an analog value ... 2324
14.2.1 Description of Limit ... 2324
14.2.2 Limit modes ... 2326
14.2.3 Limit functions ... 2326
14.2.4 Limit error handling ... 2327
14.2.5 Limit messaging ... 2328
14.2.6 Limit I/Os ... 2328
14.2.7 Limit block diagram ... 2329

14.3 MuxAn03 - Selection of an analog value to increase availability / safety ... 2330
14.3.1 Description of MuxAn03 ... 2330
14.3.2 MuxAn03 modes ... 2331
14.3.3 MuxAn03 functions ... 2331
14.3.4 MuxAn03 error handling ... 2334
14.3.5 MuxAn03 messaging ... 2335
14.3.6 MuxAn03 I/Os ... 2335
14.3.7 MuxAn03 block diagram ... 2337

14.4 MuxAn08 - Selection of an analog value to increase availability / safety ... 2338
14.4.1 Description of MuxAn08 ... 2338
14.4.2 MuxAn08 modes ... 2338
14.4.3 MuxAn08 functions ... 2339
14.4.4 MuxAn08 error handling ... 2341
14.4.5 MuxAn08 messaging ... 2341
14.4.6 MuxAn08 I/Os ... 2341
14.4.7 MuxAn08 block diagram ... 2343

14.5 RateLim - Signal ramp ... 2344

14.5.1 Description of RateLim ... 2344
14.5.2 RateLim modes ... 2345
14.5.3 RateLim functions ... 2345
14.5.4 RateLim error handling ... 2348
14.5.5 RateLim messaging ... 2348
14.5.6 RateLim I/Os ... 2349
14.5.7 RateLim block diagram ... 2351

14.6 RedAn02 - 1 out of 2 selection for redundant analog values ... 2352
14.6.1 Description of RedAn02 ... 2352
14.6.2 RedAn02 modes ... 2353
14.6.3 RedAn02 functions ... 2353
14.6.4 RedAn02 error handling ... 2354
14.6.5 RedAn02 messaging ... 2354
14.6.6 RedAn02 I/Os ... 2354
14.6.7 RedAn02 block diagram ... 2356

14.7 SelA02In - Output of two analog values ... 2357
14.7.1 Description of SelA02In ... 2357
14.7.2 SelA02In modes ... 2358
14.7.3 SelA02In functions ... 2358
14.7.4 SelA02In error handling ... 2359
14.7.5 SelA02In messaging ... 2360
14.7.6 SelA02In I/Os ... 2360
14.7.7 SelA02In block diagram ... 2361

14.8 SelA16In - Output of 16 analog values ... 2362
14.8.1 Description of SelA16In ... 2362
14.8.2 SelA16In modes ... 2363
14.8.3 SelA16In functions ... 2364
14.8.4 SelA16In error handling ... 2367
14.8.5 SelA16In messaging ... 2368
14.8.6 SelA16In I/Os ... 2368
14.8.7 SelA16In block diagram ... 2372
14.8.8 Operator control and monitoring ... 2373
14.8.8.1 SelA16In views ... 2373
14.8.8.2 SelA16In standard view ... 2374
14.8.8.3 SelA16In preview ... 2376
14.8.8.4 Block icon for SelA16In ... 2377

14.9 SelAn16L - Large version of SelAn16In ... 2380
14.9.1 Description of SelAn16L ... 2380
14.9.2 SelAn16L modes ... 2382
14.9.3 SelAn16L functions ... 2382
14.9.4 SelAn16L error handling ... 2385
14.9.5 SelAn16L messaging ... 2386
14.9.6 SelAn16L I/Os ... 2386
14.9.7 SelAn16L block diagram ... 2392
14.9.8 Operaror control and monitoring ... 2393
14.9.8.1 SenAn16L views ... 2393
14.9.8.2 SelAn16L standard view ... 2394
14.9.8.3 SelAn16L preview ... 2395
14.9.8.4 Block icon for SelAn16L ... 2397

**15 Digital logic blocks ..... 2399**

15.1 And04 - Forming an AND signal from 4 binary input signals ..... 2399
15.1.1 Description of And04 ..... 2399
15.1.2 And04 modes ..... 2400
15.1.3 And04 functions ..... 2400
15.1.4 And04 error handling ..... 2401
15.1.5 And04 messaging ..... 2401
15.1.6 And04 I/Os ..... 2401
15.1.7 And04 block diagram ..... 2402

15.2 And08 - Forming an AND signal from 8 binary input signals ..... 2404
15.2.1 Description of And08 ..... 2404
15.2.2 And08 modes ..... 2405
15.2.3 And08 functions ..... 2405
15.2.4 And08 error handling ..... 2405
15.2.5 And08 messaging ..... 2406
15.2.6 And08 I/Os ..... 2406
15.2.7 And08 block diagram ..... 2408

15.3 FlipFlop - Flip Flop with Clock Input ..... 2409
15.3.1 Description of FlipFlop ..... 2409
15.3.2 FlipFlop modes ..... 2412
15.3.3 FlipFlop functions ..... 2412
15.3.4 FlipFlop error handling ..... 2413
15.3.5 FlipFlop messaging ..... 2414
15.3.6 FlipFlop I/Os ..... 2414
15.3.7 FlipFlop block diagram ..... 2416

15.4 Or04 - Forming an OR signal from 4 binary input signals ..... 2417
15.4.1 Description of Or04 ..... 2417
15.4.2 Or04 modes ..... 2418
15.4.3 Or04 functions ..... 2419
15.4.4 Or04 error handling ..... 2419
15.4.5 Or04 messaging ..... 2420
15.4.6 Or04 I/Os ..... 2420
15.4.7 Or04 block diagram ..... 2421

15.5 Or08 - Forming an OR signal from 8 binary input signals ..... 2422
15.5.1 Description of Or08 ..... 2422
15.5.2 Or08 modes ..... 2423
15.5.3 Or08 functions ..... 2423
15.5.4 Or08 error handling ..... 2424
15.5.5 Or08 messaging ..... 2424
15.5.6 Or08 I/Os ..... 2425
15.5.7 Or08 block diagram ..... 2426

15.6 Not01 - Inversion of an input signal ..... 2427
15.6.1 Description of Not01 ..... 2427
15.6.2 Not01 modes ..... 2428
15.6.3 Not01 functions ..... 2428
15.6.4 Not01 error handling ..... 2428
15.6.5 Not01 messaging ..... 2429
15.6.6 Not01 I/Os ..... 2429
15.6.7 Not01 block diagram ..... 2430

15.7 RedDi02 - 1 out of 2 selection for redundant digital values ... 2431
15.7.1 Description of RedDi02 ... 2431
15.7.2 RedDi02 modes ... 2432
15.7.3 RedDi02 functions ... 2432
15.7.4 RedDi02 error handling ... 2433
15.7.5 RedDi02 messaging ... 2433
15.7.6 RedDi02 I/Os ... 2433
15.7.7 RedDi02 block diagram ... 2435

15.8 SelD02In - Output of one of two digital signals ... 2436
15.8.1 Description of SelD02In ... 2436
15.8.2 SelD02In modes ... 2437
15.8.3 SelD02In functions ... 2437
15.8.4 SelD02In error handling ... 2438
15.8.5 SelD02In messaging ... 2438
15.8.6 SelD02In I/Os ... 2439
15.8.7 SelD02In block diagram ... 2440

15.9 StrctCom - Structure composer for 32-bit structures ... 2441
15.9.1 Description of StrctCom ... 2441
15.9.2 StrctCom modes ... 2442
15.9.3 StrctCom functions ... 2442
15.9.4 StrctCom error handling ... 2443
15.9.5 StrctCom messaging ... 2443
15.9.6 StrctCom I/Os ... 2443
15.9.7 StrctCom block diagram ... 2444

15.10 StrctDeC - Structure decomposer for 32-bit structures ... 2446
15.10.1 Description of StrctDeC ... 2446
15.10.2 StrctDeC modes ... 2447
15.10.3 StrctDeC functions ... 2447
15.10.4 StrctDeC error handling ... 2448
15.10.5 StrctDeC messaging ... 2448
15.10.6 StrctDeC I/Os ... 2448
15.10.7 StrctDeC block diagram ... 2449

15.11 Trigger - Detection of rising and falling edges ... 2451
15.11.1 Description of Trigger ... 2451
15.11.2 Trigger modes ... 2452
15.11.3 Trigger functions ... 2453
15.11.4 Trigger error handling ... 2453
15.11.5 Trigger messaging ... 2454
15.11.6 Trigger I/Os ... 2454
15.11.7 Trigger block diagram ... 2455

15.12 XOr04 - EXCLUSIVE OR logic operation ... 2456
15.12.1 Description of XOr04 ... 2456
15.12.2 XOr04 modes ... 2457
15.12.3 XOr04 functions ... 2457
15.12.4 XOr04 error handling ... 2458
15.12.5 XOr04 messaging ... 2458
15.12.6 XOr04 I/Os ... 2459
15.12.7 XOr04 block diagram ... 2460

**16 Generator blocks ..... 2461**

16.1 NoiseGen - Generating signal noise ..... 2461
16.1.1 Description of NoiseGen ..... 2461
16.1.2 NoiseGen I/Os ..... 2461

**17 Channel blocks ..... 2463**

17.1 Information on using channel blocks ..... 2463

17.2 FbAnIn - Analog input channel block for field devices ..... 2466
17.2.1 Description of FbAnIn ..... 2466
17.2.2 FbAnIn modes ..... 2468
17.2.3 FbAnIn functions ..... 2468
17.2.4 FbAnIn error handling ..... 2471
17.2.5 FbAnIn messaging ..... 2472
17.2.6 FbAnIn I/Os ..... 2472
17.2.7 FbAnIn block diagram ..... 2475

17.3 FbAnOu - Analog output channel block for field devices ..... 2476
17.3.1 Description of FbAnOu ..... 2476
17.3.2 FbAnOu modes ..... 2478
17.3.3 FbAnOu functions ..... 2478
17.3.4 FbAnOu error handling ..... 2480
17.3.5 FbAnOu messaging ..... 2480
17.3.6 FbAnOu I/Os ..... 2481
17.3.7 FbAnOu block diagram ..... 2486

17.4 FbAnTot - Analog totalizer channel block for field devices ..... 2487
17.4.1 Description of FbAnTot ..... 2487
17.4.2 FbAnTot modes ..... 2488
17.4.3 FbAnTot functions ..... 2488
17.4.4 FbAnTot error handling ..... 2490
17.4.5 FbAnTot messaging ..... 2491
17.4.6 FbAnTot I/Os ..... 2491
17.4.7 FbAnTot block diagram ..... 2494

17.5 FbDiIn - Digital input channel block for field devices ..... 2495
17.5.1 Description of FbDiIn ..... 2495
17.5.2 FbDiIn modes ..... 2496
17.5.3 FbDiIn functions ..... 2497
17.5.4 FbDiIn error handling ..... 2499
17.5.5 FbDiIn messaging ..... 2500
17.5.6 FbDiIn I/Os ..... 2500
17.5.7 FbDiIn block diagram ..... 2503

17.6 FbDiOu - Digital output channel block for field devices ..... 2505
17.6.1 Description of FbDiOu ..... 2505
17.6.2 FbDiOu modes ..... 2506
17.6.3 FbDiOu functions ..... 2507
17.6.4 FbDiOu error handling ..... 2509
17.6.5 FbDiOu messaging ..... 2510
17.6.6 FbDiOu I/Os ..... 2510
17.6.7 FbDiOu block diagram ..... 2515

17.7 FbDrive - Channel block for compact drives ..... 2516

17.7.1 Description of FbDrive ... 2516
17.7.2 FbDrive modes ... 2517
17.7.3 FbDrive functions ... 2517
17.7.4 FbDrive error handling ... 2519
17.7.5 FbDrive messaging ... 2519
17.7.6 FbDrive status word ... 2520
17.7.7 FbDrive I/Os ... 2520
17.7.8 FbDrive block diagram ... 2527

17.8 FbEnMe - Channel block for ET 200SP Energy Meter ... 2528
17.8.1 Description of FbEnMe ... 2528
17.8.2 FbEnMe modes ... 2529
17.8.3 FbEnMe functions ... 2530
17.8.4 FbEnMe error handling ... 2531
17.8.5 FbEnMe messaging ... 2532
17.8.6 FbEnMe I/Os ... 2532
17.8.7 FbEnMe block diagram ... 2539

17.9 FbSwtMMS - Channel block for MM starter ... 2541
17.9.1 Description of FbSwtMMS ... 2541
17.9.2 FbSwtMMS modes ... 2542
17.9.3 FbSwtMMS functions ... 2542
17.9.4 FbSwtMMS error handling ... 2543
17.9.5 FbSwtMMS messaging ... 2544
17.9.6 FbSwtMMS status word ... 2544
17.9.7 FbSwtMMS I/Os ... 2545
17.9.8 FbSwtMMS block diagram ... 2550

17.10 Pcs7AITC - Analog input driver, dynamic measuring range center ... 2551
17.10.1 Pcs7AITC description ... 2551
17.10.2 Pcs7AITC functions ... 2554
17.10.3 Pcs7AITC error handling ... 2558
17.10.4 Pcs7AITC messaging ... 2559
17.10.5 Pcs7AITC I/Os ... 2559

17.11 Pcs7AnIn - Analog input channel block ... 2564
17.11.1 Description of Pcs7AnIn ... 2564
17.11.2 Pcs7AnIn modes ... 2566
17.11.3 Pcs7AnIn functions ... 2567
17.11.4 Pcs7AnIn error handling ... 2571
17.11.5 Pcs7AnIn messaging ... 2572
17.11.6 Pcs7AnIn I/Os ... 2572
17.11.7 Pcs7AnIn block diagram ... 2576

17.12 Pcs7AnOu - Analog output channel block ... 2577
17.12.1 Description of Pcs7AnOu ... 2577
17.12.2 Pcs7AnOu modes ... 2580
17.12.3 Pcs7AnOu functions ... 2580
17.12.4 Pcs7AnOu error handling ... 2582
17.12.5 Pcs7AnOu messaging ... 2583
17.12.6 Pcs7AnOu I/Os ... 2584
17.12.7 Pcs7AnOu block diagram ... 2587

17.13 Pcs7DiCnt - Digital input driver with count functionality ... 2588
17.13.1 Pcs7DiCnt description ... 2588

17.13.2 Pcs7DiCnt functions ... 2589
17.13.3 Pcs7DiCnt error handling ... 2591
17.13.4 Pcs7DiCnt messaging ... 2591
17.13.5 Pcs7DiCnt I/Os ... 2592

17.14 Pcs7DiIn - Digital input channel block ... 2596
17.14.1 Description of Pcs7DiIn ... 2596
17.14.2 Pcs7DiIn modes ... 2598
17.14.3 Pcs7DiIn functions ... 2598
17.14.4 Pcs7DiIn error handling ... 2600
17.14.5 Pcs7DiIn messaging ... 2601
17.14.6 Pcs7DiIn I/Os ... 2601
17.14.7 Pcs7DiIn block diagram ... 2604

17.15 Pcs7DiIT - Digital input channel block with time stamp ... 2605
17.15.1 Description of Pcs7DiIT ... 2605
17.15.2 Pcs7DiIT modes ... 2607
17.15.3 Pcs7DiIT functions ... 2607
17.15.4 Pcs7DiIT error handling ... 2610
17.15.5 Pcs7DiIT messaging ... 2610
17.15.6 Pcs7DiIT I/Os ... 2611
17.15.7 Pcs7DiIT block diagram ... 2614

17.16 Pcs7DiOu - Digital output channel block ... 2616
17.16.1 Description of Pcs7DiOu ... 2616
17.16.2 Pcs7DiOu modes ... 2618
17.16.3 Pcs7DiOu functions ... 2619
17.16.4 Pcs7DiOu error handling ... 2620
17.16.5 Pcs7DiOu messaging ... 2621
17.16.6 Pcs7DiOu I/Os ... 2621
17.16.7 Pcs7DiOu block diagram ... 2624

17.17 Pcs7Cnt1 - Controlling and reading FM 350 modules ... 2625
17.17.1 Description of Pcs7Cnt1 ... 2625
17.17.2 Pcs7Cnt1 modes ... 2627
17.17.3 Pcs7Cnt1 functions ... 2627
17.17.4 Pcs7Cnt1 error handling ... 2629
17.17.5 Pcs7Cnt1 messaging ... 2630
17.17.6 Pcs7Cnt1 I/Os ... 2630
17.17.7 Pcs7Cnt1 block diagram ... 2635

17.18 Pcs7Cnt2 - Control and read an 8-DI_NAMUR module of the ET 200iSP ... 2636
17.18.1 Description of Pcs7Cnt2 ... 2636
17.18.2 Pcs7Cnt2 modes ... 2639
17.18.3 Pcs7Cnt2 functions ... 2639
17.18.4 Pcs7Cnt2 error handling ... 2641
17.18.5 Pcs7Cnt2 messaging ... 2642
17.18.6 Pcs7Cnt2 I/Os ... 2642
17.18.7 Pcs7Cnt2 block diagram ... 2645

17.19 Pcs7Cnt3 - Control and read the 1 COUNT 24V/100kHz module for count mode ... 2646
17.19.1 Description of Pcs7Cnt3 ... 2646
17.19.2 Pcs7Cnt3 modes ... 2648
17.19.3 Pcs7Cnt3 functions ... 2648
17.19.4 Pcs7Cnt3 error handling ... 2649

17.19.5 Pcs7Cnt3 messaging ... 2650
17.19.6 Pcs7Cnt3 I/Os ... 2650
17.19.7 Pcs7Cnt3 block diagram ... 2654

17.20 Pcs7Cnt4 - Handling counting functionality of TM modules ... 2655
17.20.1 Description of Pcs7Cnt4 ... 2655
17.20.2 Pcs7Cnt4 modes ... 2658
17.20.3 Pcs7Cnt4 functions ... 2658
17.20.4 Pcs7Cnt4 error handling ... 2659
17.20.5 Pcs7Cnt4 messaging ... 2660
17.20.6 Pcs7Cnt4 I/Os ... 2660
17.20.7 Pcs7Cnt4 block diagram ... 2664

17.21 Pcs7HaAI - HART variable channel block for AI-HART modules ... 2665
17.21.1 Description of Pcs7HaAI ... 2665
17.21.2 Pcs7HaAI modes ... 2666
17.21.3 Pcs7HaAI functions ... 2666
17.21.4 Pcs7HaAI error handling ... 2667
17.21.5 Pcs7HaAI messaging ... 2668
17.21.6 Pcs7HaAI I/Os ... 2668
17.21.7 Pcs7HaAI block diagram ... 2671

17.22 Pcs7HaAO - HART variable channel block for AO-HART modules ... 2672
17.22.1 Description of Pcs7HaAO ... 2672
17.22.2 Pcs7HaAO modes ... 2673
17.22.3 Pcs7HaAO functions ... 2673
17.22.4 Pcs7HaAO error handling ... 2674
17.22.5 Pcs7HaAO messaging ... 2675
17.22.6 Pcs7HaAO I/Os ... 2675
17.22.7 Pcs7HaAO block diagram ... 2678

17.23 Pcs7ReIn - REAL input channel block ... 2679
17.23.1 Description of PCS7ReIn ... 2679
17.23.2 PCS7ReIn modes ... 2680
17.23.3 PCS7ReIn functions ... 2680
17.23.4 PCS7ReIn error handling ... 2682
17.23.5 PCS7ReIn messaging ... 2683
17.23.6 PCS7ReIn I/Os ... 2683
17.23.7 PCS7ReIn block diagram ... 2686

17.24 Annex for channel blocks ... 2687
17.24.1 Mode Settings for SM Modules ... 2687
17.24.2 Mode settings for field devices ... 2698
17.24.3 Mode settings for Channel blocks ... 2700

**18 Conversion blocks ... 2701**

18.1 StrgToBy - String in byte structure (Struct of Byte) ... 2701
18.1.1 Description of StrgToBy ... 2701
18.1.2 StrgToBy modes ... 2702
18.1.3 StrgToBy functions ... 2702
18.1.4 StrgToBy error handling ... 2703
18.1.5 StrgToBy messaging ... 2703
18.1.6 StrgToBy I/Os ... 2703
18.1.7 StrgToBy block diagram ... 2704

18.2 StruAnIn - Separating an analog structured variable ... 2706
18.2.1 Description of StruAnIn ... 2706
18.2.2 StruAnIn modes ... 2707
18.2.3 StruAnIn functions ... 2707
18.2.4 StruAnIn error handling ... 2707
18.2.5 StruAnIn messaging ... 2708
18.2.6 StruAnIn I/Os ... 2708
18.2.7 StruAnIn block diagram ... 2709

18.3 StruAnOu - Creating an analog structured variable ... 2710
18.3.1 Description of StruAnOu ... 2710
18.3.2 StruAnOu modes ... 2711
18.3.3 StruAnOu functions ... 2711
18.3.4 StruAnOu error handling ... 2711
18.3.5 StruAnOu messaging ... 2712
18.3.6 StruAnOu I/Os ... 2712
18.3.7 StruAnOu block diagram ... 2713

18.4 StruDiIn - Separating a digital structured variable ... 2714
18.4.1 Description of StruDiIn ... 2714
18.4.2 StruDiIn modes ... 2715
18.4.3 StruDiIn functions ... 2715
18.4.4 StruDiIn error handling ... 2715
18.4.5 StruDiIn messaging ... 2716
18.4.6 StruDiIn I/Os ... 2716
18.4.7 StruDiIn block diagram ... 2717

18.5 StruDiOu - Creating a digital structured variable ... 2718
18.5.1 Description of StruDiOu ... 2718
18.5.2 StruDiOu modes ... 2719
18.5.3 StruDiOu functions ... 2719
18.5.4 StruDiOu error handling ... 2719
18.5.5 StruDiOu messaging ... 2720
18.5.6 StruDiOu I/Os ... 2720
18.5.7 StruDiOu block diagram ... 2721

18.6 StruScIn - Separating a display area into two variables ... 2722
18.6.1 Description of StruScIn ... 2722
18.6.2 StruScIn modes ... 2723
18.6.3 StruScIn functions ... 2723
18.6.4 StruScIn error handling ... 2723
18.6.5 StruScIn messaging ... 2724
18.6.6 StruScIn I/Os ... 2724
18.6.7 StruScIn block diagram ... 2725

18.7 StruScOu - Merging two variables into a display area ... 2726
18.7.1 Description of StruScOu ... 2726
18.7.2 StruScOu modes ... 2727
18.7.3 StruScOu functions ... 2727
18.7.4 StruScOu error handling ... 2727
18.7.5 StruScOu messaging ... 2728
18.7.6 StruScOu I/Os ... 2728
18.7.7 StruScOu block diagram ... 2729

18.8 STIn - Separating the signal status into individual binary displays ... 2730

18.8.1 Description of STIn ... 2730
18.8.2 STIn modes ... 2731
18.8.3 STIn functions ... 2731
18.8.4 STIn error handling ... 2731
18.8.5 STIn messaging ... 2732
18.8.6 STIn I/Os ... 2732
18.8.7 STIn block diagram ... 2733

18.9 STOu - Merging individual binary signals into a signal status ... 2734
18.9.1 Description of STOu ... 2734
18.9.2 STOu modes ... 2735
18.9.3 STOu functions ... 2735
18.9.4 STOu error handling ... 2735
18.9.5 STOu messaging ... 2736
18.9.6 STOu I/Os ... 2736
18.9.7 STOu block diagram ... 2737

18.10 MSTIn - Separating the maintenance status into individual status displays ... 2738
18.10.1 Description of MSTIn ... 2738
18.10.2 MSTIn modes ... 2739
18.10.3 MSTIn functions ... 2739
18.10.4 MSTIn error handling ... 2739
18.10.5 MSTIn messaging ... 2740
18.10.6 MSTIn I/Os ... 2740
18.10.7 MSTIn block diagram ... 2741

18.11 MSTOu - Merging individual status displays into a maintenance status ... 2742
18.11.1 Description of MSTOu ... 2742
18.11.2 MSTOu modes ... 2743
18.11.3 MSTOu functions ... 2743
18.11.4 MSTOu error handling ... 2744
18.11.5 MSTOu messaging ... 2744
18.11.6 MSTOu I/Os ... 2744
18.11.7 MSTOu block diagram ... 2745

18.12 RealToDw - Converting REAL to DWORD ... 2747
18.12.1 Description of RealToDw ... 2747
18.12.2 RealToDw modes ... 2748
18.12.3 ReaLToDw functions ... 2748
18.12.4 RealToDw error handling ... 2748
18.12.5 RealToDw messaging ... 2748
18.12.6 RealToDw I/Os ... 2749
18.12.7 RealToDw block diagram ... 2749

18.13 StateMap - Conversion of other signal states into APL signal states ... 2750
18.13.1 Description of StateMap ... 2750
18.13.2 StateMap modes ... 2751
18.13.3 StateMap functions ... 2751
18.13.4 StateMap error handling ... 2751
18.13.5 StateMap messaging ... 2751
18.13.6 StateMap I/Os ... 2751
18.13.7 StateMap block diagram ... 2752

18.14 StruCmIn - Splitting a structure to outputs ... 2753
18.14.1 Description of StruCmIn ... 2753

18.14.2 StruCmIn modes ... 2753
18.14.3 StruCmIn functions ... 2754
18.14.4 StruCmIn error handling ... 2754
18.14.5 StruCmIn messaging ... 2754
18.14.6 StruCmIn I/Os ... 2754
18.14.7 StruCmIn block diagram ... 2755

18.15 StruCmOu - Merging inputs to a structure ... 2756
18.15.1 Description of StruCmOu ... 2756
18.15.2 StruCmOu modes ... 2756
18.15.3 StruCmOu functions ... 2757
18.15.4 StruCmOu error handling ... 2757
18.15.5 StruCmOu messaging ... 2757
18.15.6 StruCmOu I/Os ... 2757
18.15.7 StruCmOu block diagram ... 2758

18.16 StruCnIn - Separating a counter structured variable ... 2759
18.16.1 Description of StruCnIn ... 2759
18.16.2 StruCnIn modes ... 2759
18.16.3 StruCnIn functions ... 2760
18.16.4 StruCnIn error handling ... 2760
18.16.5 StruCnIn messaging ... 2760
18.16.6 StruCnIn I/Os ... 2760
18.16.7 StruCnIn block diagram ... 2761

18.17 StruCnOu - Creating a structured counter variable ... 2762
18.17.1 Description of StruCnOu ... 2762
18.17.2 StruCnOu modes ... 2762
18.17.3 StruCnOu functions ... 2763
18.17.4 StruCnOu error handling ... 2763
18.17.5 StruCnOu messaging ... 2763
18.17.6 StruCnOu I/Os ... 2763
18.17.7 StruCnOu block diagram ... 2764

**19 Maintenance blocks ... 2765**

19.1 MuxMST - Determination of the worst maintenance status ... 2765
19.1.1 Description of MuxMST ... 2765
19.1.2 MuxMST modes ... 2766
19.1.3 MuxMST functions ... 2766
19.1.4 MuxMST error handling ... 2767
19.1.5 MuxMST messaging ... 2767
19.1.6 MuxMST I/Os ... 2767
19.1.7 MuxMST block diagram ... 2768

19.2 MuxST- Determination of the worst signal status ... 2770
19.2.1 Description of MuxST ... 2770
19.2.2 MuxST modes ... 2771
19.2.3 MuxST functions ... 2771
19.2.4 MuxST error handling ... 2772
19.2.5 MuxST messaging ... 2772
19.2.6 MuxST I/Os ... 2773
19.2.7 MuxST block diagram ... 2774

19.3 STRep - Status display of block groups ... 2775
19.3.1 Description of STRep ... 2775

19.3.2 STRep modes ... 2776
19.3.3 STRep functions ... 2776
19.3.4 STRep error handling ... 2776
19.3.5 STRep messaging ... 2776
19.3.6 STRep I/Os ... 2776
19.3.7 STRep block diagram ... 2777

19.4 AssetM - Process variable monitoring for violation of limits ... 2778
19.4.1 Description of AssetM ... 2778
19.4.2 AssetM modes ... 2781
19.4.3 AssetM functions ... 2781
19.4.4 AssetM error handling ... 2782
19.4.5 AssetM messages ... 2782
19.4.6 AssetM I/Os ... 2783
19.4.7 AssetM block diagram ... 2785

**20 System blocks ... 2787**

20.1 AddInt64 - Addition of two 64-bit integer variables ... 2787
20.1.1 Description of AddInt64 ... 2787

20.2 AddR64 - Addition of two 64-bit REAL variables ... 2788
20.2.1 Description of AddR64 ... 2788

20.3 DiToInt64 - Converting from DINT to Int64 ... 2789
20.3.1 Description of DiToInt64 ... 2789

20.4 Int64ToDi - Converting from Int64 to DINT ... 2790
20.4.1 Description of Int64ToDi ... 2790

20.5 MemR256 - Increasing the number of internal previous values ... 2791
20.5.1 Description of MemR256 ... 2791

20.6 NegInt64 - Negation of an Int64 variable ... 2792
20.6.1 Description of NegInt64 ... 2792

20.7 NegR64 - Negation of a Real64 variable ... 2793
20.7.1 Description of NegR64 ... 2793

20.8 PIDCoefR - Calculation of coefficients ... 2794
20.8.1 Description of PIDCoefR ... 2794

20.9 R64ToReal - Converting Real64 to REAL ... 2795
20.9.1 Description of R64ToReal ... 2795

20.10 RealToR64 - Converting REAL to Real64 ... 2796
20.10.1 Description of RealToR64 ... 2796

20.11 SelST16 - Output of the best or worst signal status ... 2797
20.11.1 Description of SelST16 ... 2797

20.12 ShLeInt64 - Left shift of an Int64 variable ... 2798
20.12.1 Description of ShLeInt64 ... 2798

20.13 ShRiInt64 - Right shift of an Int64 variable ... 2799
20.13.1 Description of ShRiInt64 ... 2799

20.14 PIDKernR - Calculation of the manipulated variable ... 2800
20.14.1 Description of PIDKernR ... 2800

**21 Communication blocks ..... 2801**

21.1 Snd_DigVal - Send Boolean values including quality code (Snd_DigVal) ..... 2801
21.1.1 Description of Snd_DigVal ..... 2801
21.1.2 Snd_DigVal modes ..... 2802
21.1.3 Snd_DigVal functions ..... 2802
21.1.4 Snd_DigVal error handling ..... 2803
21.1.5 Snd_DigVal messages ..... 2803
21.1.6 Snd_DigVal I/Os ..... 2803
21.1.7 Snd_DigVal block diagram ..... 2804

21.2 Rcv_DigVal - Receive Boolean values including quality code (Rcv_DigVal) ..... 2805
21.2.1 Description of Rcv_DigVal ..... 2805
21.2.2 Rcv_DigVal modes ..... 2806
21.2.3 Rcv_DigVal functions ..... 2807
21.2.4 Rcv_DigVal error handling ..... 2807
21.2.5 Rcv_DigVal messages ..... 2807
21.2.6 Rcv_DigVal I/Os ..... 2807
21.2.7 Rcv_DigVal block diagram ..... 2808

21.3 Snd_AnaVal - Send analog values including quality code (SND_AnaVal) ..... 2809
21.3.1 Description of Snd_AnaVal ..... 2809
21.3.2 Snd_AnaVal modes ..... 2810
21.3.3 Snd_AnaVal functions ..... 2810
21.3.4 Snd_AnaVal error handling ..... 2810
21.3.5 Snd_AnaVal messages ..... 2811
21.3.6 Snd_AnaVal I/Os ..... 2811
21.3.7 Snd_AnaVal block diagram ..... 2811

21.4 Rcv_AnaVal - Receive analog values including quality code (Rcv_AnaVal) ..... 2812
21.4.1 Description of Rcv_AnaVal ..... 2812
21.4.2 Rcv_AnaVal modes ..... 2813
21.4.3 Rcv_AnaVal functions ..... 2813
21.4.4 Rcv_AnaVal error handling ..... 2814
21.4.5 Rcv_AnaVal messages ..... 2814
21.4.6 Rcv_AnaVal I/Os ..... 2814
21.4.7 Rcv_AnaVal block diagram ..... 2815

**22 Process tag types (insertible templates) ..... 2817**

22.1 Introduction to process tag types ..... 2817

22.2 PID controller (PIDControl_Lean) ..... 2820

22.3 PID controller for PA/FF devices (PIDControl_Lean_Fb) ..... 2821

22.4 PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) ..... 2822

22.5 PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) ..... 2823

22.6 PID controller with safety logic and manipulated variable ramp (PIDConR_MV_Ramp) ..... 2824

22.7 PID - control with operating-point-oriented parameter control (GainScheduling) ..... 2825

22.8 PID controller with dynamic feedforward control (FfwdDisturbCompensat) ..... 2827

22.9 PID controller with Smith predictor (SmithPredictorControl) ..... 2830

22.10 Step controller with direct access to the actuator and without position feedback (StepControlDirect) ... 2831
22.11 Step controller with assigned actuator block and position feedback (StepControlActor) ... 2832
22.12 Split-range controller with control loop monitoring through ConPerMon (SplitrangeControl) ... 2833
22.13 Split-Range control (SplitRangeControl_Lean) ... 2835
22.14 Ratio control with control loop monitoring through ConPerMon (RatioControl) ... 2836
22.15 Ratio control (RatioControl_Lean) ... 2838
22.16 Ratio control with PIDConR (RatioR) ... 2839
22.17 Cascade control with control loop monitoring through ConPerMon (CascadeControl) ... 2840
22.18 Cascade control (CascadeControl_Lean) ... 2842
22.19 Cascade control with PIDConR (CascadeR) ... 2843
22.20 Cascade control with PIDStepL (CascadeStepControl) ... 2844
22.21 Source chart for GainSched function block (gain scheduling) ... 2845
22.22 Override control ... 2846
22.23 Override control with PIDConR (OverrideR) ... 2848
22.24 Model-based predictive control (ModPreCon) ... 2849
22.25 Monitoring of a digital process tag (DigitalMonitoring) ... 2851
22.26 Monitoring a digital process tag for PA/FF devices (DigitalMonitoring_Fb) ... 2852
22.27 Monitoring eight digital process tags (Digital8Monitoring) ... 2853
22.28 Monitoring an analog process tag (AnalogMonitoring) ... 2854
22.29 Monitoring of an analog process tag for PA/FF devices (AnalogMonitoring_Fb) ... 2855
22.30 Dosing (Dose_Lean) ... 2856
22.31 Dosing with PA/FF devices (Dose_Lean_Fb) ... 2857
22.32 Motor (Motor_Lean) ... 2858
22.33 Motor with PROFIdrive Drive Profile telegram 1 and 20 (Namur) ... 2859
22.34 Two-speed motor (Motor2Speed) ... 2860
22.35 Reversing motor (MotorReversible) ... 2861
22.36 Reversible motor with controllable speed (MotorSpeedControlled) ... 2862
22.37 Motor with an additional analog value and time-stamped signals (Motor_AV_EventTs) ... 2863
22.38 Motor according to the profile for low voltage switchgear devices with profile 1 of the MM_Starter ... 2864
22.39 Valve (Valve_Lean) ... 2865
22.40 Two-way valve (Valve2Way) ... 2866
22.41 Motor valve (ValveMotor) ... 2867
22.42 Control valve (VlvAnL) ... 2868

22.43 Control valve for PA/FF devices (ValveAnalog_Fb) ..... 2869
22.44 Example project APL_Example_xx ..... 2870
22.44.1 Introduction to the PCS 7 example project for Advanced Process Control ..... 2870
22.44.2 Process simulation including noise generator (ProcSimC; ProcSimS) ..... 2871
22.44.3 Cascade control of temperature by using the heat flow (CascadeSim) ..... 2873
22.44.4 Control loop monitoring for simulation with colored noise (ConPerMonSim) ..... 2875
22.44.5 Feedforward control to compensate a measurable disturbance variable (DisturbCompSim) ..... 2875
22.44.6 Operating point-oriented adaptation of parameters (gain scheduling) for non-linear processes (GainSchedSim) ..... 2876
22.44.7 Override control on a pipeline (OverrideSim) ..... 2877
22.44.8 Smith predictor for a dead time system (SmithPredictorSim) ..... 2877
22.44.9 Filtering of noisy measured values in a control loop (SigSmoothSim) ..... 2877
22.44.10 Predictive control of a 2x2 multi-variable controlled system (ModPreConSim) ..... 2879
22.44.11 Predictive control of a non-linear process (ModPreConNonLinSim) ..... 2879

**23 Definitions ..... 2881**

23.1 Batch process ..... 2881
23.2 Approximation ..... 2882
23.3 Prediction horizon ..... 2883
23.4 Trajectory ..... 2884
23.5 Maverick ..... 2885
23.6 Ergodic process ..... 2886
23.7 Conti process ..... 2887
23.8 Multivariable controller ..... 2888
23.9 non-phase minimum ..... 2889

**Index ..... 2891**

# Security information

# 1

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
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
https://www.siemens.com/industrialsecurity.

# Basics of APL

# 2

## 2.1 Functions of the blocks

### 2.1.1 General information

#### 2.1.1.1 User-configured message classes

**User-configured message classes**

The message classes Alarm, Warning and Tolerance, the corresponding abbreviations in symbols and colors, and the terms used in this documentation are not valid for user-configured message classes.

These terms and colors depend on the project-specific setting. The user-configured messages classes are only supported for block symbols as of V9.

#### 2.1.1.2 Forcing operating modes

**Forcing operating modes**

The "forcing of operating modes" function lets you set the function block into a different operating mode using interconnectable input parameters, regardless of the currently active control. This can, for example, be:

- Forces tracking for closed-loop controllers and control valves
- Enabling and disabling at motors
- Opening and closing of valves

It is only possible to force operating modes with "Large" blocks in the following operating modes:

- Manual mode
- Automatic mode
- Local mode "1" and "3" (only if `Feature2.Bit8` = 1)

Forcing operating modes has the highest priority over all three operating modes.

**Forcing operating modes at closed-loop controllers**

In control engineering, this procedure is also known as forced tracking of values. Refer to the Tracking and limiting a manipulated variable (Page 222) section for more on this.

### Forcing operating modes at motors and valves

The input parameter `xxxxForce = 1` (for example `OpenForce` and `CloseForce` at a valve) is used for forced controlling of the function block and thus an intervention in the function of the block, irrespective of currently active controls, interlock conditions and monitoring errors. If the input parameters are inconsistent (for example `OpenForce = 1` and `CloseForce = 1` at valves), an error number (Page 128) is output at the parameter `ErrorNum` and the control remains unchanged.

**Note**

If you have set the parameters for the advance warning time `WarnTiMan` and the idle time `IdleTime` to values higher than 0, the control will only take effect once the set times have elapsed.

**Note**

With block VlvAnl, the warning time is ignored in tracking `MV_TrkOn` = 1 and in forced tracking `MV_ForOn`.

The Enabling direct changeover between forward and reverse (Page 156) `Feature` bit has no effect when forcing the operating modes of the MotRevL and MotSpdCL blocks. Direct switchover between forward and reverse is always possible.

### Display in the faceplate and in the block icon

If an operating mode is forced, this is displayed in the block icon and in the standard view of the faceplate:

**Block icon**: In the block icon, the display for motors, valves and dosers involves the use of a red F and a crossed-out padlock.

There is no display for closed-loop controllers.

**Faceplate:** An information text on the forced operating mode is displayed in the standard view of the faceplate, for example, "Forced stop" for motors. This is also indicated by a crossed-out padlock:



### Messaging

No messages are assigned to the forcing of operating modes. However, if you want to have corresponding messages, you can use the freely interconnectable input parameters to generate the messages. Refer also to the Generating instance-specific messages (Page 230) section for more on this.

### 2.1.1.3 Resetting the block in case of interlocks or errors

#### Resetting the block

The block must be reset when an interlock has been set via the `Protect` input ("Protection"), Trip ("Protection") or an error has occurred ("Runtime" or "Control", external error `FaultExt` or `CSF` with `Feature.Bit18`).

**Note**

"Small" blocks do not feature protection (`Protect`).

The `RdyToReset` output signals when a reset can be carried out via the `RstLi` input parameter or the automatic commands.

There are different ways to reset the block:

- Reset by interconnection (input `RstLi`).
- Reset by the operator using a button in the faceplate (input `RstOp`).
- Reset with a 0-1 edge transition in the corresponding automatic or local signal (except with motor protection). Refer to the following sections for more information.

**Note**

The reset via input RstLi or RstOp does not depend on the selected operating mode.

The operator must have the appropriate authorization to use the reset function in the faceplate (`OS_Perm`). After a reset, the output parameter `P_Rst` is set for a cycle.

**Note**

**Note for VlvMotL and VlvPosL**

After resetting the block, in the case of interlocks or errors, the valve position will always be tracked, depending on `FbkOpen` and `FbkClose`.

### Resetting monitoring errors and interlocks in manual and automatic mode

You can influence the reaction using the following `Feature` Bits:

- `Feature.Bit9`: Resetting via input signals in the event of interlocking (Protection) or errors (Page 174)
- `Feature.Bit30`: Set reset depending on the operating mode or the LiOp parameter (Page 174)
- `Feature.Bit31`: Activating reset of protection / error in manual mode (Page 175)

**Note**

With `MonSafePos = 0`, a reset is not required, the valve and motor can be driven despite feedback failure.

### Resetting monitoring errors, external errors and interlocks in local mode

The monitoring error can occur in local mode if you have set 1, 3 or 5 for the input parameter `LocalSetting` (see Local mode (Page 86)). When `LocalSetting` is set to 2 or 4, a monitoring error can only occur when a rapid stop is triggered.

The following applies with `LocalSetting` 1 or 3:

The monitoring error, the external error and the interlocks cannot be reset when the control and feedback signals do not match.

With `Feature2.Bit13 = 0`: the following applies:

- When the control and feedback signals match, the monitoring error, external error and the interlocks are reset by stopping (`StopLocal = 1`) the drive.
- With `Vlv2WayL` in the `MonSafePos = 1` setting, a monitoring error is reset by `Pos0Local = 1`.
- With `VlvL`, `VlvMotL` in the `MonSafePos = 1` setting, a monitoring error is reset with the local command, which moves the valve to the neutral position.
- With `Vlv2WayL`, `VlvMotL` and `VlvL` in the `MonSafePos = 0` setting, no reset of the monitoring error is required. The currently pending control is in effect.
- With `Vlv2WayL`, `VlvL` and `VlvMotL`, an external error is reset with the local command, which moves the valve to the neutral position
- With `DoseL`, you must acknowledge the protection (`Protect`) and flow alarms with a positive edge at the "`CancelLocal`" or "`PauseLocal`" output parameter.
- With MotL, MotRevL, MotSpdCL or MotSpdL in the MonSafePos =1 setting, an external or monitoring error or an interlock protection is reset with the local stop command.

With `Feature2.Bit13 = 1`: The reset can be done from the reset button in the standard view of the faceplate.

The following applies with `LocalSetting` 2, 4 or 5:

No reset required.

### Resetting motor protection (Trip) in local mode

There are two ways for resetting the Motor protection In Local mode 1 & 3. Once the trip is healthy and feedback is withdrawn then:

- If `Feature2.Bit13 = 0` the reset of pending errors will not be possible via faceplate.
- If `Feature2.Bit13 = 1` then reset button will be enabled in the standard view of the faceplate and operator can reset using this button.

### Resetting monitoring errors, external errors and interlocks using the "Forcing operating states" function

With "Forcing operating states", monitoring errors, external errors, interlocks or the motor protection function are reset under the following conditions and a reset pulse is output at the `P_Rst` output:

- The block is in an operating mode in which a reset is necessary and
- a monitoring error, an external error, a "Protection" interlock or the motor protection function is ready to be reset. This can be seen in the faceplate with the reset button or with the Request 0/1 indicator in the faceplate. When `Feature.Bit19` = 1, the block is ready to reset as soon as the protection `(Protect = 0)` or motor protection `(Trip = 0)` interlock is set, whereby enabled motor protection prevents the motor from starting.

See also the following chapter: Forcing operating modes (Page 45).

### Tabular overview for resetting for interlocks and errors

| | **Permit** | **Interlock** | **Protect** | **PermitSafePos** |
|---|---|---|---|---|
| **Meaning** | Activation enable ("Permission") | Interlock without reset ("Interlock") | Interlock with reset ("Protection") | Activation enable ("PermitSafePos") |
| **Description** | The activation enable (input `Permit` = 1) makes it possible to leave the neutral position of the block in response to operator input or a command from the program (CFC/SFC). The activation enable has no effect if the block is not in the neutral position. | A pending interlock condition brings the block to the neutral position (input `Intlock` = 0). | A pending interlock condition brings the block to the neutral position (input `Protect` = 0). | The activation enable (input PermitSafePos = 0) prevents the control in neutral position or the stop command of the block in response to operator input or a command from the program (CFC/SFC). |

| | **Permit** | **Interlock** | **Protect** | **PermitSafePos** |
|---|---|---|---|---|
| **Mode:**<br>**Automatic** | Takes effect if block is in the neutral position.<br>After the interlock condition has gone, the currently pending control function becomes active again. | After the interlock condition has gone, the currently pending control function becomes active again. | `Feature.Bit9` **and** `Feature.Bit30` **= 0:** Reset via faceplate or `RstLi` = 1<br>`Feature.Bit9` **= 1 and** `Feature.Bit30` **= 0:** Reset via faceplate or `RstLi` = 1 or a 0-1 edge transition in the control<br>`Feature.Bit9` **= 0 and** `Feature.Bit30` **= 1:** Reset via `RstLi` = 1<br>`Feature.Bit9` **and** `Feature.Bit30` **= 1:** Reset via `RstLi = 1` or a 0-1 edge transition in the control = 1 or 0-1 edge transition in the control | After the interlock permission condition has gone, the currently pending control function becomes active again. |

| | **Permit** | **Interlock** | **Protect** | **PermitSafePos** |
|---|---|---|---|---|
| **Mode:**<br>**Local** | Takes effect if block is in the neutral position.<br>After the interlock condition has gone, the currently pending control function becomes active again. | After the interlock condition has gone, the currently pending control function becomes active again. | The following applies with `LocalSetting` = 1 or 3:<br>**Generally:** When the control and feedback signals match, reset via `StopLocal` = 1.<br>**Vlv2WayL, VlvMotL und VlvL:** Reset via local command, which moves the valve into the neutral position.<br>**DoseL:** Reset via a positive edge at "`CancelLocal`" or "`PauseLocal`".<br>The following applies with `LocalSetting` = 2,4 or 5:<br>No reset required | - |
| **Mode:**<br>**Manual** | Takes effect if block is in the neutral position.<br>It is possible to leave the neutral position with an operation in the faceplate. | The faceplate can be operated again after the interlock condition has gone. | `Feature.Bit30` **and** `Feature.Bit31` **= 0:** Resetting not necessary<br>`Feature.Bit30` **= 1 and** `Feature.Bit31` **= 0:** Resetting not necessary<br>`Feature.Bit30` **= 0 and** `Feature.Bit31` **= 1:** Reset via faceplate or `RstLi` = 1<br>`Feature.Bit30` **and** `Feature.Bit31` **= 1:** Reset via faceplate | After the interlock permission condition has gone, the currently pending control function becomes active again. |

| | **Trip** | **Error** | **Rapid stop** |
|---|---|---|---|
| **Meaning** | Motor protection | Monitoring errors and external errors | Rapid stop |
| **Description** | The motor protection function is used to switch off the motor when there is a heat overload (input `Trip` = 0). | • Monitoring the startup and stop characteristics for motors or the runtime of valves<br>• Monitoring the operation of motors or the maintenance of the position of valves<br>• External error FaultExt: Block goes to error state without a message being output.<br>• External control system fault CSF with set `Feature.Bit18`: block reports an external control system fault and goes to error state. | A rapid stop stops the drive immediately. |
| **Mode:**<br>**Automatic** | `Feature.Bit9` **and** `Feature.Bit30` **= 0:** Reset via faceplate or `RstLi` = 1<br>`Feature.Bit9` **= 1 and** `Feature.Bit30` **= 0:** Reset via faceplate or `RstLi` = 1 or a 0-1 edge transition in the control<br>`Feature.Bit9` **= 0 and** `Feature.Bit30` **= 1:** Reset via `RstLi` = 1<br>`Feature.Bit9` **and** `Feature.Bit30` **= 1:** Reset via `RstLi = 1` or a 0-1 edge transition in the control = 1 or 0-1 edge transition in the control | | `Feature.Bit9` **= 0:** Reset via faceplate or `RstLi = 1`<br>`Feature.Bit9` **= 1:** Reset via faceplate or `RstLi` = 1 or a 0-1 edge transition in the control |

| | Trip | Error | Rapid stop |
|---|---|---|---|
| **Mode:**<br>**Local** | The following applies with LocalSetting = 1 or 3:<br>When the control and feedback signals of the drive match, reset via StopLocal = 1.<br>The following applies with LocalSetting = 2, 4 or 5:<br>No reset required.<br>For more information, please refer Resetting the Pending Error in Local Mode 1 and 3 (Page 210). | The following applies with LocalSetting = 1 or 3:<br>• When the control and feedback signals of the drive match, reset via StopLocal = 1.<br>• With Vlv2WayL, VlvMotL, VlvPosL, VlvL, MotL, MotRevL, MotSpdCL, MotSpdL.<br>– **Monitoring error with** MonSafePos = **1:** Reset via the local command, which moves the valve into the neutral position or the motor in stop position.<br>– **Monitoring error with** MonSafePos = 0**:** No resetting required; the currently pending control function is active.<br>– **External error:** Reset via the local command, which moves the valve into the neutral position.<br>For more information, refer Resetting the Pending Error in Local Mode 1 and 3 (Page 210).<br>• With DoseL, resetting via a positive edge at "CancelLocal" or "PauseLocal".<br>The following applies with LocalSetting = 2, 4 or 5:<br>No reset required. | The rapid stop function is unlocked in the faceplate via the "Reset" button (RstOp = 1). In CFC, unlocking is carried out using the input parameter RstLi = 1<br>For more information, please refer Resetting the Pending Error in Local Mode 1 and 3 (Page 210). |
| **Mode:**<br>**Manual** | Feature.Bit30 **and** Feature.Bit31 **= 0:** Resetting not necessary<br>Feature.Bit30 **= 1 and** Feature.Bit31 **= 0:** Resetting not necessary<br>Feature.Bit30 **= 0 and** Feature.Bit31 **= 1** Reset via faceplate or RstLi = 1<br>Feature.Bit30 **and** Feature.Bit31 **= 1:** Reset via faceplate | | The rapid stop function is unlocked in the faceplate via the "Reset" button (RstOp = 1). In CFC, unlocking is carried out using the input parameter RstLi = 1 |

### 2.1.1.4 Neutral position for motors, valves and controllers

#### Neutral position for motors, valves and controllers

The neutral position always represents the deenergized state.

#### Neutral position for motors

The neutral position for motors is always the stopped motor.

The neutral position is adopted when:

- The function Monitoring the feedbacks (Page 104) was triggered.
- One of the interlock conditions is active (see Interlocks (Page 107)).
- An external error via `FaultExt` or `CSF` was triggered (see Error handling (Page 128)).
- The Motor protection function (Page 106) was triggered.
- The function Rapid stop for motors (Page 115) was triggered.
- During start-up (see Set startup characteristics (Page 148)).
- The "Out of service" mode is active.
- One of the automatic commands has bad signal status (16#00 or 16#28) and the `Feature2` bit Considering bad quality of automatic commands or external values (Page 200) is set to 1.

#### Neutral position for valves

There are different forms of the deenergized state for valves:

- Valve is closed in a de-energized state
- Valve is open in a de-energized state
- Valve is stopped in a de-energized state (e.g. motor valve)

The input parameter `SafePos` is used to set these properties of the valve:

- `SafePos = 0`: Valve is closed in a de-energized state
- `SafePos = 1`: Valve is open in a de-energized state
- `SafePos = 2`: Valve is stopped in a de-energized state (e.g. motor valve)

The neutral position is adopted when:

- The function Monitoring the feedbacks (Page 104) was triggered.
- One of the interlock conditions is active (see Interlocks (Page 107)).
- An external error via `FaultExt` or `CSF` was triggered (see Error handling (Page 128)).
- During start-up (see Set startup characteristics (Page 148)).
- The "Out of service" mode is active.
- One of the automatic commands has bad signal status (16#00 or 16#28) and the `Feature2` bit Considering bad quality of automatic commands or external values (Page 200) is set to 1.

### Neutral position for the VlvAnL and VlvPosL blocks (actuator)

The possible neutral positions are set by the SafePos parameter:

- SafePos = 0: Neutral position of the control valve is "Closed" (MV.Value = MV_LoLim)
- SafePos = 1: Neutral position of the control valve is "Open" (MV.Value = MV_HiLim)
- SafePos = 2: Neutral position of the control valve is "Stop" (MV.Value = RbkOut, if Rbk.ST <> 16#FF, else MV.Value remains unchanged)

The control valve VlvAnL is brought to the neutral position when the FbkAuxVCloseOut = 1 auxiliary valve is closed.

The neutral position of the control valve VlvAnL and VlvPosL is adopted when:

- The function Monitoring the feedbacks (Page 104) was triggered.
- One of the interlock conditions is active (see Interlocks (Page 107)).
- An external error via FaultExt or CSF was triggered (see Error handling (Page 128)).
- During start-up (see Set startup characteristics (Page 148)).
- The "Out of service" mode is active.
- One of the automatic commands has bad signal status (16#00 or 16#28) and the Feature2 bit Considering bad quality of automatic commands or external values (Page 200) is set to 1.

In case of startup or “Out of service” mode, the valve VlvAnL has additional Feature bits for adopting the neutral position:

- During start-up, if the Feature bit Set startup characteristics (Page 148) and the Feature bit Neutral position manipulated variable takes effect at startup (Page 176) are set.
- In the "Out of service" mode, if the Feature bit Neutral position manipulated variable takes effect with "out of service" operating mode (Page 177) is set.

### Neutral position for continuous controllers (does not apply to controller modules)

- Only the limits or a special analog value for the manual value are taken into consideration for the neutral position with continuous controllers.
  The input parameter SafePos is used to specify the neutral position:
  - SafePos = 0 corresponds to the low limit (ManLoLim or MV_LoLim for PIDConS)
  - SafePos = 1 corresponds to the high limit (ManHiLim or MV_HiLim for PIDConS)
- The input parameters SafePos and SafePos2 are used to determine the neutral position:

| SafePos | SafePos2 | Neutral position | Remarks |
|---|---|---|---|
| =0 | =0 | ManLoLim for PIDConL, PIDConR or MV_LoLim for PIDConS | Low limit for manipulated variable in manual mode |
| =1 | =0 | ManHiLim for PIDConL, PIDConR or MV_HiLim for PIDConS | High limit for manipulated variable in manual mode |
| x | =1 | MV_SafePos | Manipulated variable neutral position |
| x | =2 | Last manipulated variable (stop) | Last manipulated variable (stop) |

x: Not relevant

The neutral position is adopted:

- The interlock conditions is active (see Interlocks (Page 107) only at PIDConL and PIDConR).

In case of startup or "Out of service" mode, the controller has additional `Feature` bits for adopting the neutral position:

- During start-up, if the `Feature` bit Set startup characteristics (Page 148) and the `Feature` bit Neutral position manipulated variable takes effect at startup (Page 176) are set.
- In the "Out of service" mode, if the `Feature` bit Neutral position manipulated variable takes effect with "out of service" operating mode (Page 177) is set.

#### Neutral position for step controllers (does not apply to controller modules)

You can use the input parameter `SafePos` to determine if the step controller should close, open or stop the valve when it enters the neutral position:

`SafePos = 0`: close valve

`SafePos = 1`: open valve

`SafePos = 2`: stop valve

When the neutral position (fully opened or fully closed) is reached and a limit stop signal (`FbkOpened` or `FbkClosed`) is set, the valve is stopped (Stop = 1).

The neutral position is adopted:

- during start-up if the `Feature` bit Set startup characteristics (Page 148) and the `Feature` bit Neutral position manipulated variable takes effect at startup (Page 176) are set.
- in the "Out of service" mode if the `Feature` bit Neutral position manipulated variable takes effect with "out of service" operating mode (Page 177) is set.

#### Safety control for controllers of the FM 355 or FM 355-2 modules

The controller modules have their own mechanism for feedforwarding a safety value (see Temperature Controller FM 355-2 manual or Controller Module FM 355 manual)

### 2.1.1.5 Specifying warning times for control functions at motors and valves

#### Specifying warning times for control functions at motors and valves

This function is only supported by "Large" blocks.

You can generate warning signals when, for example, motors are started or valves are opened. Warning signals can be generated in the following modes:

- Manual mode (input parameter `WarnTiMan`)
- Automatic mode (input parameter `WarnTiAut`)

You specify the warning times in seconds using the input parameters `WarnTiMan` and `WarnTiAut`. If, for example, a motor is started then, this is displayed at the output parameter with `WarnAct = 1`. The motor then starts after the configured warning time has expired and `WarnAct` is reset (`WarnAct = 0`).

A corresponding warning is not output if the warning times (`WarnTiMan` or `WarnTiAut`) are specified with a smaller value than the `SampleTime` parameter.

**Note**

In this case, the warning time is only active if the block is controlled from the de-energized state.

### Disabling warnings

Configure each parameter with 0 seconds to generate no warnings.

## 2.1.1.6 Output signal as a static signal or pulse signal

### Output signal as a static signal or pulse signal

You can output the control signals for motors, valves and dosers as:

- Static signal or as a
- Pulse signal with configurable pulse length.

You can find the signals in the I/O table of the individual blocks.

**Note**

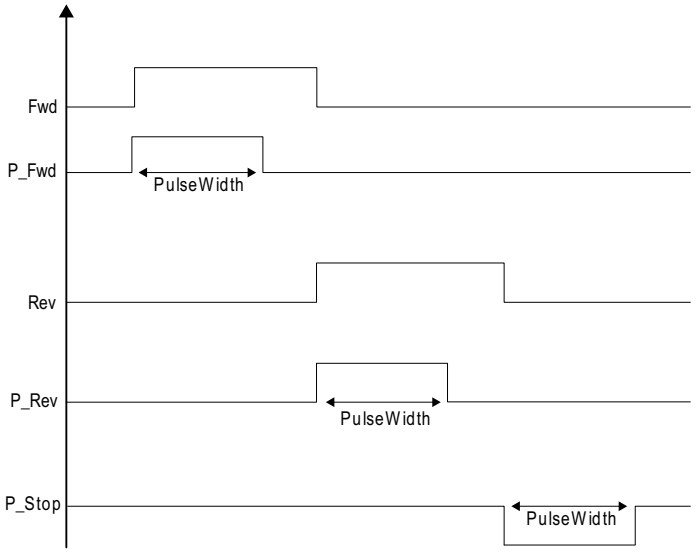
The pulse signal is available only for the "large" blocks.

### Output signal as a static signal

The control settings are made available as a static signal in the blocks in the form of interconnectable output parameters. The MotRevL block, for example, provides these as static signals via the alternative output parameters `Fwd`, `Rev` and `Run`.

### Output signal as a pulse signal

The control is made available as pulse signals at the blocks using interconnectable output parameters. You specify the pulse length of the output signals in seconds using the input parameter `PulseWidth`. The MotRevL block, for example, provides these as pulse signals via the output parameters `P_Fwd`, `P_Rev` and `P_Stop`.




**Note**

Almost all output parameters for pulse control, for example `P_Fwd`, `P_Rev`, `P_Ctrl`, have a positive effective direction, i.e. a 0→1→0 pulse triggers activation.

The only exception is the `P_Stop` output parameter with a negative effective direction, i.e. a 1→0→1 pulse triggers activation.

## 2.1.1.7 Recording the first signal for interlock blocks

### Recording the first signal

You activate the function described below using the `Feature` bit " Activating recording of the first signal (Page 161)".

The number of the input that caused the last output signal change from 1 to 0 (good state to locked) is displayed for you in bit coding at the `FirstIn` output. The cause may be:

- A signal change at the input or a change in inversion
  Example: With an OR logic operation, the single 1 changes to 0. The output then changes from 1 to 0.
- A change to the I/O
  Example: Excluding the single 1 then results in an output of 0 with an OR logic operation.

- A change to the signal status
  If the signal status of the input, which forms the output value alone and has the value 1, changes from 16#80 to 16#00, the output value changes from 1 to 0.
- `FirstIn` is not changed if the following events occur, despite a change to the output:
  - Change in output value from or to `DefaultOut`

If several signals are at the same time responsible for the change, all responsible inputs are indicated in the faceplate and output in bit coding in the `FirstIn` output. If the input signals change without this causing the signal at the output to change, `FirstIn` does not change.

Inputs which are not interconnected or which are excluded are not taken into account.

You can reset `FirstIn` to 0 if you set the `RstLi` input from 0 to 1 (positive edge) or you can operate the `RstOp` input using the faceplate ("Reset" button).

If at least one bit in `FirstIn` is set, other signal changes are not taken into account.

### 2.1.1.8 Outputting a signal for start readiness

#### Outputting a signal for start readiness

The `RdyToStart = 1` output parameter is used to indicate start readiness in automatic mode.

Start readiness is output when the following conditions are met:

- No group error pending (group fault with consideration of `Feature.Bit18` for `CSF` and `MonSafePos` for feedback errors)
- No interlock is active
- No forced operating mode or manipulated variable is active
- No rapid stop is active (only applies to motors or `VlvMotL`)
- The block is in automatic mode (not with controller blocks)
- The waiting time for the restart must have expired (only applies to motors or `VlvMotL`)

The start readiness is shown at the following block groups via the `RdyToStart` output parameter:

- Motors
- Valves
- Dosers
- Software controller
- Hardware controller

### Start readiness for motors

The start readiness for motor blocks is formed as follows:

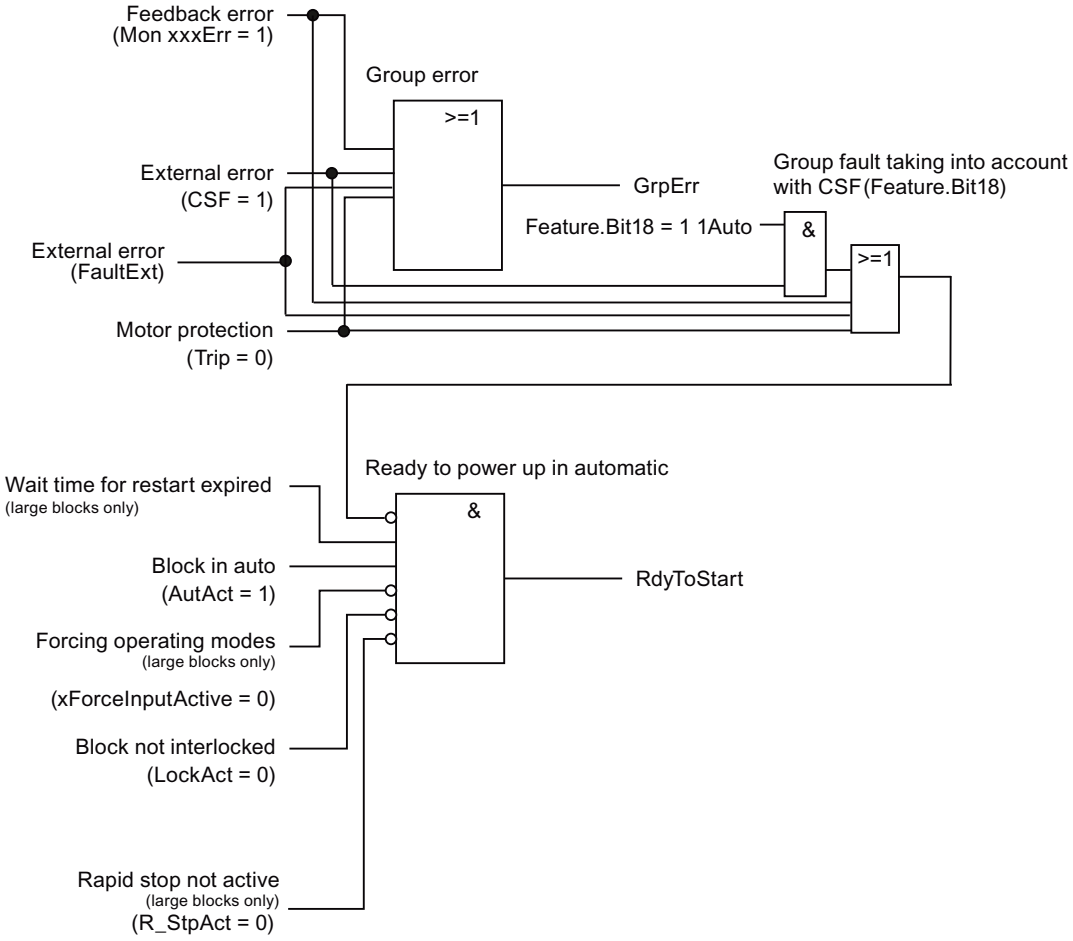



Figure 2-1 Output signal for start readiness for motors

## Start readiness for valves

The start readiness for valve blocks is formed as follows:

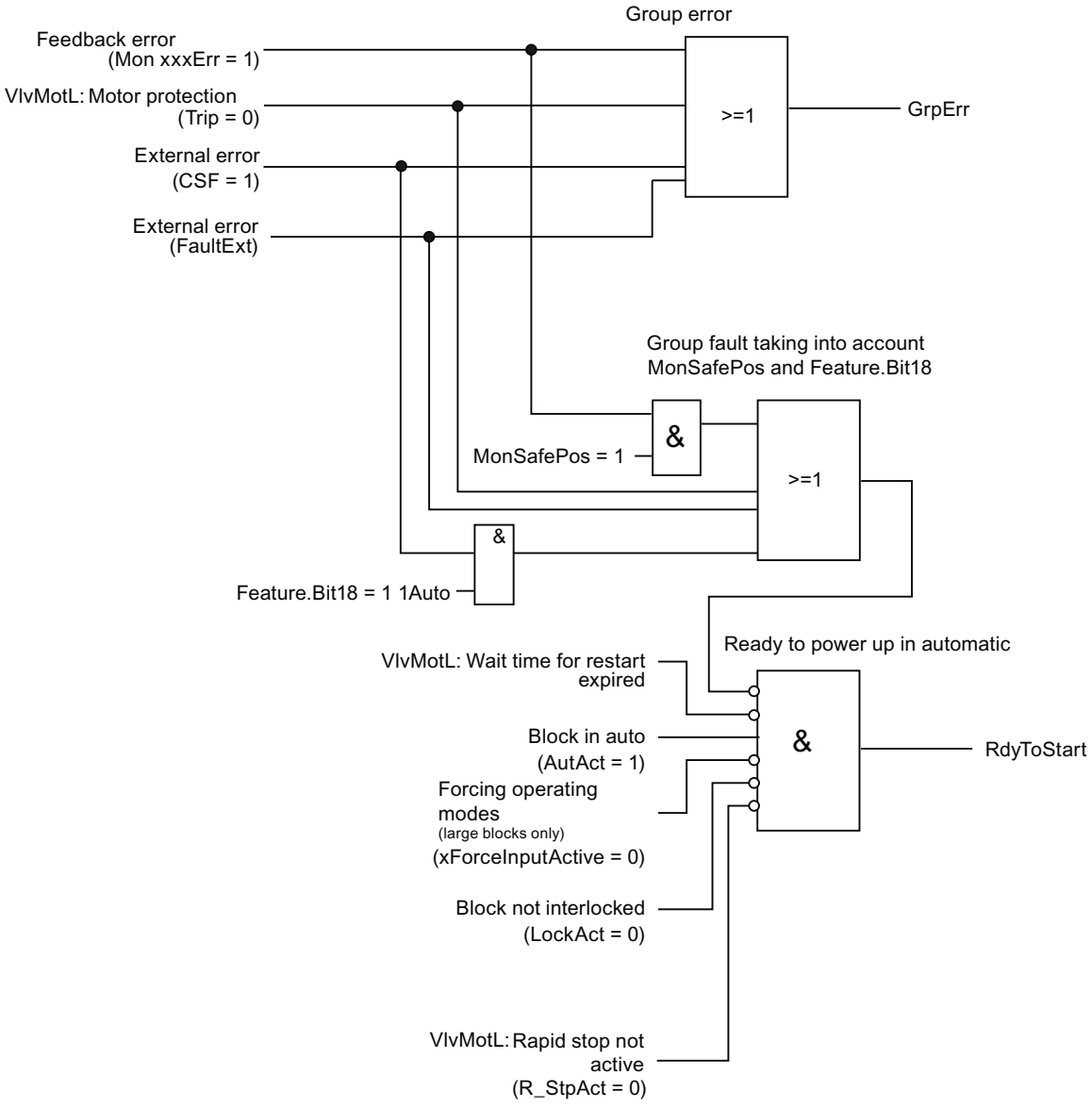



Figure 2-2 Output signal for start readiness for valves

### Start readiness for dosers

The start readiness for dosers is formed as follows:

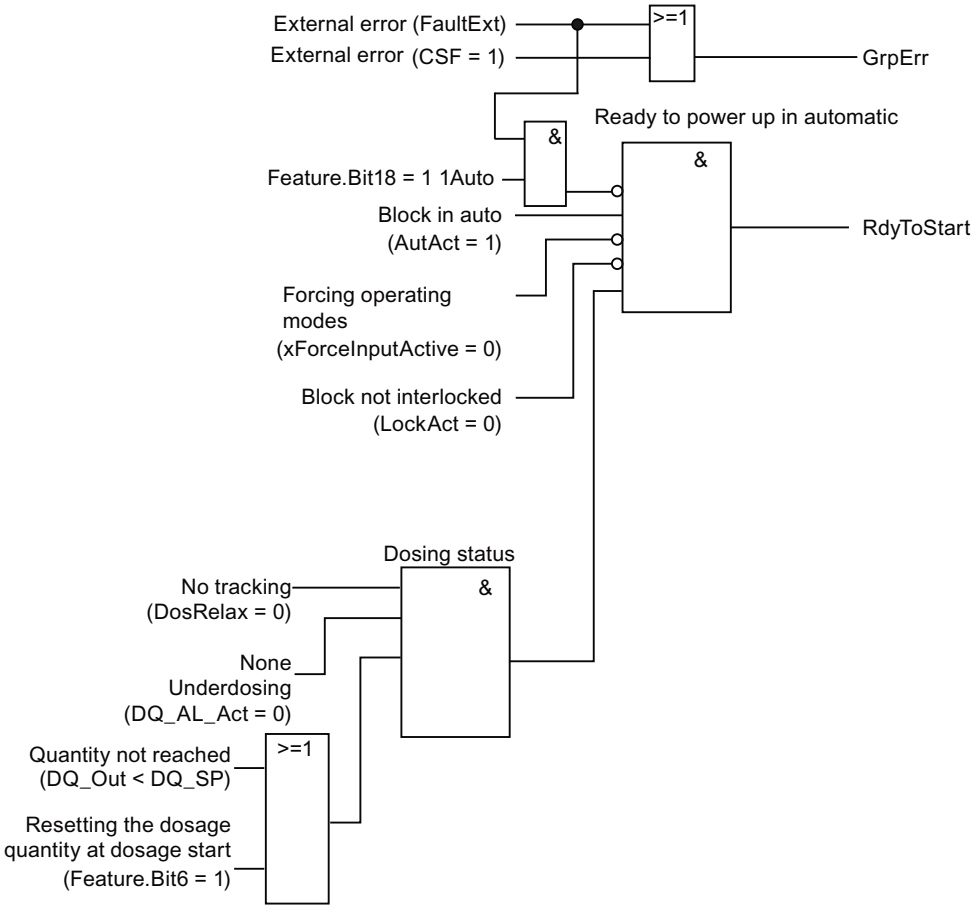



Figure 2-3 Output signal for start readiness for dosers

### Start readiness for software controllers

The start readiness for software controllers is formed as follows:

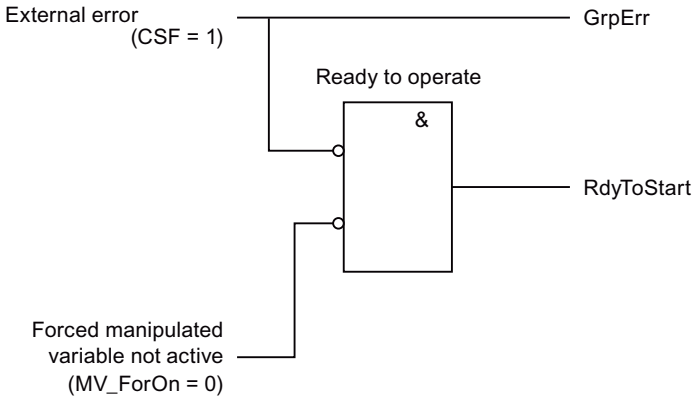



Figure 2-4 Output signal for start readiness for software controllers

### Start readiness for hardware controllers

The start readiness for hardware controllers is formed as follows:

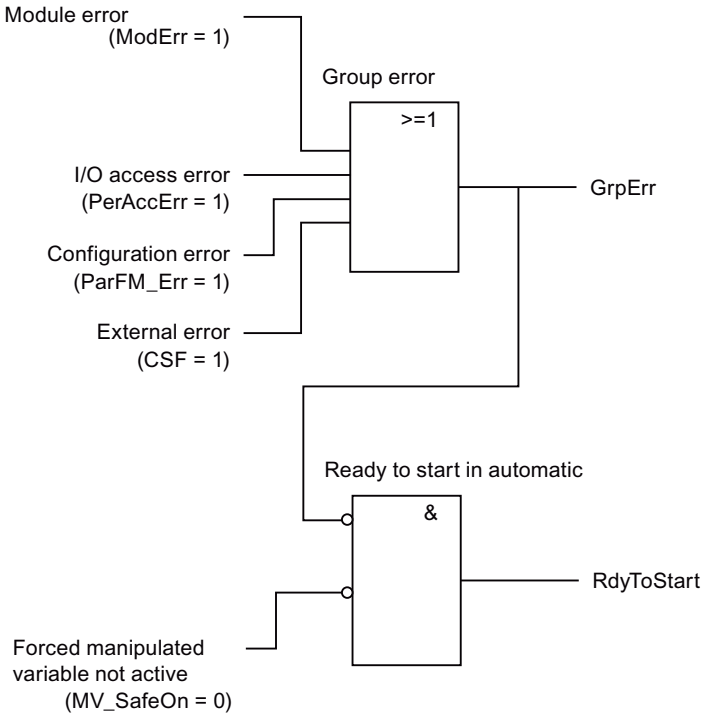



Figure 2-5 Output signal for start readiness for hardware controllers

## 2.1.1.9 Simulating signals

### Simulating signals

Simulation means the manipulation of a signal regardless of the actual source of the signal or logic that generates this signal.

Simulation is carried out either at the field device (externally from the control system) itself or at a block (internally in the control system).

In either case, the associated status in the signal is set to the simulation value (see also Forming and outputting signal status for blocks (Page 116)).

During the simulation, every block is considered in isolation. There are two different forms of simulation here, namely:

- Block-external simulation and
- Block-internal simulation.

### Block-external simulation

Block-external simulation is characterized by the fact that:

- The simulation function is not executed in the block itself and
- A signal whose status has the simulation state, for example, a simulation of the signal at another block or directly in the I/O device, is applied at an input parameter.

The block-external simulation has the following effects on the functionality of the block:

- The technological functions are not influenced
- All the process-relevant output signals do not receive the simulation status. In the case of technologic blocks, process-related output signals are parameters that actively affect the process, e.g. "`Start`" for block MotL.
- In the case of blocks with operator control or monitoring functions (for example faceplates), these signals are identified in the faceplate with the status for the simulation as follows:



- Blocks with one or more input parameters for signals with "Generate status from individual status" receive a group status in accordance with the priority table. This group status is displayed in the status bar of the block icon and of the faceplate with the simulation status as follows:



- The interlocking functions of the block are not influenced.

**Note**

For the output channel blocks, you need to specify the exact block response with external simulation using the `Feature` bit Outputting a de-energized value for block-external simulation (Page 159).

### Block-internal simulation

Block-internal simulation is characterized by the "simulation" function being run in the block itself.

With operator control and monitoring blocks, all process values that cannot be controlled (e.g. `PV`, `AV`, `In`) can be simulated. This is used primarily as an aid for commissioning and servicing of the system. For example, the control settings of a motor can be simulated and the feedback values corrected without the monitoring functions being active.

For specific technological blocks (e.g. large blocks) that can be operated and monitored, simulation can take place via the faceplate as well as interconnectable inputs:

- `SimLiOp` = 0: The simulation is activated/deactivated via faceplate (parameter view) at the input `SimOn`.

- `SimLiOp` = 1: The simulation is activated/deactivated via the input `SimOnLi`. The interconnectable simulation values (e.g. `SimPV_Li`, `SimAV_Li`, `SimInLi`) will become effective in the process. In this case, the inputs `SimOn`, `SimPV`, `SimAV`, `SimIn`, `SimRbk` are written back with `SimOnLi`, `SimPV_Li`, `SimAV_Li`, `SimInLi`, `SimRbkLi`.

The `Feature` bit Activating the run time of feedback signals (Page 164) can be used to delay tracking of the feedback signals for motors and valves (for example, `Fbkxxx`).

Simulation can also be carried out for blocks (such as channel blocks) that cannot be controlled and monitored by the operator.

The control is simulated in the CFC by setting parameters directly in the block with the input parameters `SimOn = 1` and `Simxxxx =` for the desired simulation value (e.g. `SimPV` , `SimAV` or `SimIn`).

**Note**

With channel blocks, ensure that the `Mode` parameter is set correctly during simulation.

**Note**

If the block is not in simulation, the simulation values (`SimPV`, `SimAV`, `SimIn`, `SimRbk`) are updated to the process values (`PV`, `AV`, `In`, `Rbk`).

Simulation is triggered during runtime in the faceplate's parameter view by clicking on the "Simulation" button.

This simulation is characterized by the fact that:

- The simulation can only be enabled / disabled with the operator authorization level for system authorization.
- The technological functions are not influenced.
- All the process-related output signals receive the "simulation" status. In the case of technologic blocks, process-related output signals are parameters that actively affect the process, e.g. "`Start`" for block MotL.
- In the case of blocks with operator control or monitoring functions (for example faceplates), these signals are identified in the faceplate with the status for the simulation as follows:

- The group status of the block is displayed in the status bar of the block icon and of the faceplate with the simulation status as follows:

- All the process values displayed in the faceplate that cannot be operated-controlled in normal operation (e.g. `PV`).
- When the block control can be manipulated, the readback and feedback values (for example `Rbk`, `FbkSpd1`) are adjusted according to the manipulation of the control.
- Associated values (for example `UserAna1`) cannot be simulated.
- The interlocking functions of the block are activated in accordance with input parameter `BypProt = 0` or deactivated (`BypProt = 1`). This is shown as follows in the faceplate and block icon:

| | |
|---|---|
| Enabled |  |
| Disabled |  |

### Block-internal simulation for controllers

How block-internal simulation for controllers works (`SimOn = 1`):

- In manual mode, both the simulated process value `SimPV` and the simulated repeated manipulated variable `SimRbk` can be entered in the faceplate as a simulated value.
- When a switchover to automatic mode is performed, the simulated process value `SimPV` is set so that it is equal to the setpoint `SP` (= tracking). This means the control deviation is no longer present and the pending manipulated variable (from the bumpless manual-automatic switchover, for example) remains constant. `SimRbk` can still be controlled.

**Note**

If you switch a controller block to block-internal simulation during automatic mode and the controller is connected to the actual process on the actuator side, you will open the control loop as a result.

The actuating signals calculated on the basis of the simulated actual value are switched to the process, but the resulting motion in the process is no longer visible in terms of the controller actual value, as a copy of the setpoint is present at this point instead, where it takes the form of a simulated actual value. The process could move away from the setpoint without the controller doing anything to counteract this and without you seeing this happen in the controller faceplate.

Manipulated variable step changes occur during switchover to automatic mode if an error signal was already present before the switchover.

The following applies to program mode:

- Program mode with setpoint specification should be considered the same as automatic mode from a control engineering point of view. Block-internal simulation reacts in the same way as it does in automatic mode: The process value `PV` is set so that it is equal to the setpoint `SP`, which in this case is derived from the input parameter `AdvCoMV`.
- Program mode with manipulated variable specification should be considered the same as manual mode from a control engineering point of view. Block-internal simulation reacts in the same way as it does in manual mode: The simulated process value `SimPV` can be entered as a simulated value. In this case the manipulated variable `MV` is derived from the input parameter `AdvCoMV`.

## 2.1.1.10 Dead band

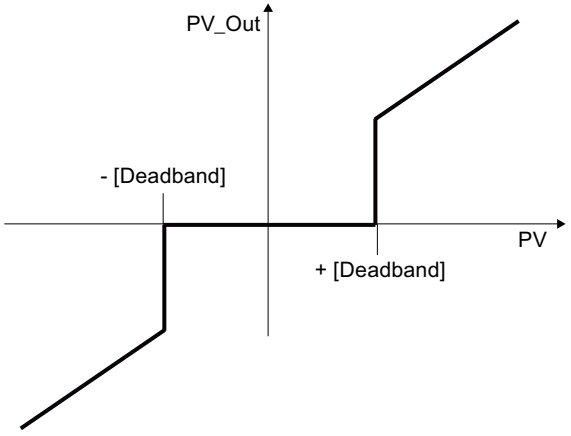
### Dead band

To suppress values fluctuating around zero, you can set a dead band (`Deadband`):

`Deadband = 0`: Dead band is disabled

`Deadband ≠ 0`: Dead band is enabled

With a negative dead band (`Deadband < 0.0`), calculation is continued internally with this value

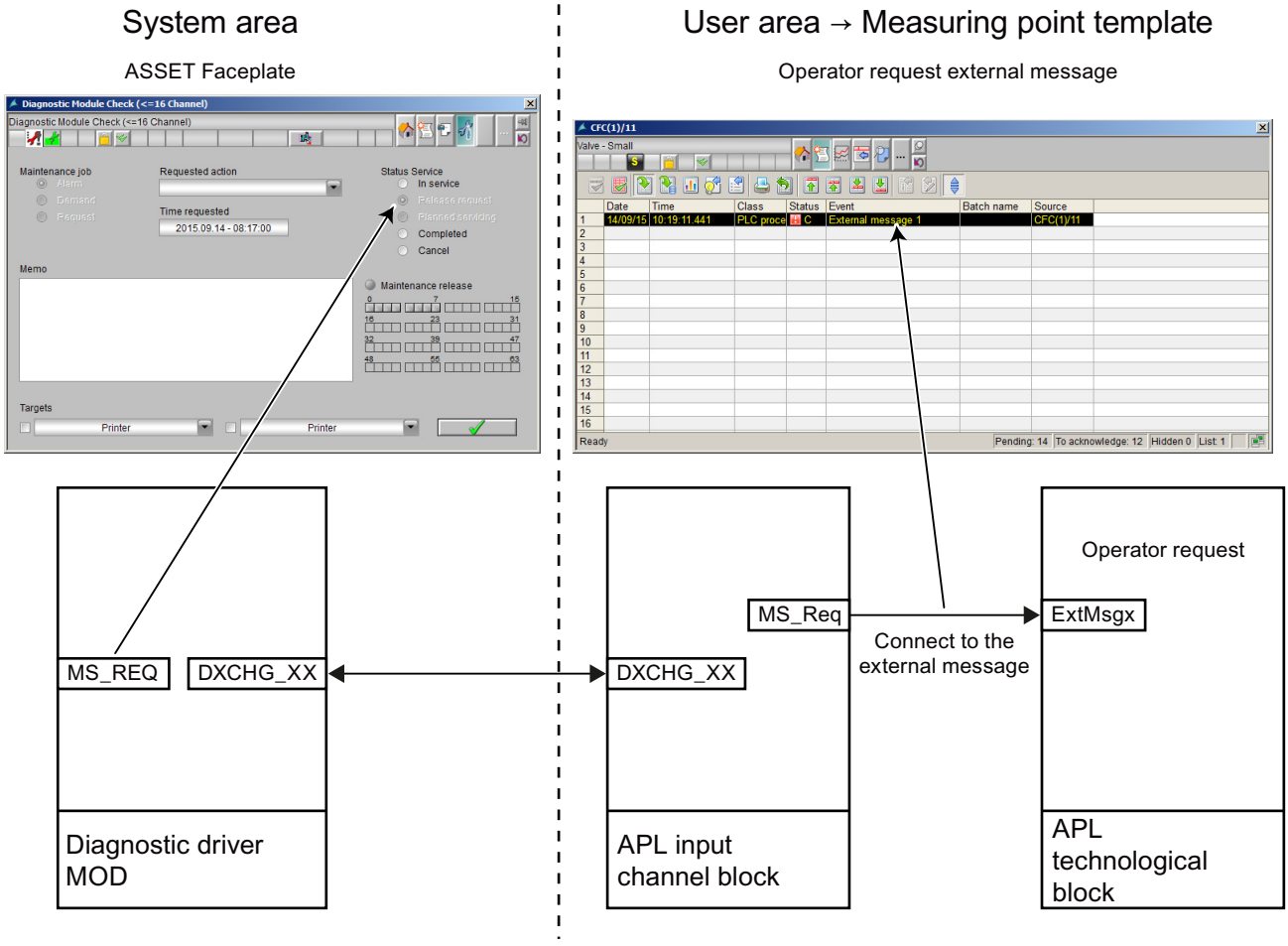



### 2.1.1.11 Request for maintenance release

#### Issuing a request for maintenance (MS → APL)

An operator can issue a request for maintenance release from the Maintenance view of the ASSET faceplate ("Status service" → "Release request"). After issuing a maintenance request, the operator can set the maintenance release. This will display the message which the operator has configured. Later, operator can set the Maintenance Release in the technological Block (Page 70). It indicates that the maintenance operator can start the maintenance.

## Functional sequence in APL




- The output parameter `MS_Req` of the channel block will be set to 1 when an operator issues a maintenance request in Maintenance view of the ASSET faceplate.
- The output parameter `MS_Req` has to be connected to the input parameter `ExtMsgx` of the desired technological block for which the user wants to issue an operator request message.
- The operator has to configure the "Message class" and enter a suitable text in the "Event" field for the operator request message in the "PCS 7 Message Configuration" window of the technological block.
- With the configured message and issued maintenance release, the maintenance operator can start the maintenance.
- The operator request message stays active until the user changes to "In service" or "Cancel" mode in the Maintenance view of ASSET faceplate.
- Once the maintenance is done, the operator can change the mode to "Completed". This will deactivate the "In service" mode (`OosAct` = 0).

### 2.1.1.12 Release for maintenance

#### Issuing a release for maintenance

The release for maintenance serves as information about a process tag at which maintenance, service or calibration should be carried out. You can use the signal for release for maintenance to transfer the information about the enabling of a process tag from the OS to a Maintenance Station.

**Note**

The block must be in either "Manual," "On" or "Out of service" mode to set the release for maintenance.

You set the release for maintenance (operator control permission "System control" required) in the parameter view using the input parameter `MS_RelOp = 1`. A release for maintenance is then made available via the interconnectable output parameter `MS_Release = 1` for further processing. In order to make this information of the Maintenance Station available, you have to interconnect the output parameter `MS_Release` of the technologic block with the input parameter `MS_Release`of the corresponding channel block.

The issuing of a release for maintenance does not have any influence on the function of the block. An operation message is generated.

#### Use of the state "In progress" on the Maintenance Station

The status "In progress" is implemented on the Maintenance Station for a process tag or a field device using the channel blocks and the interconnectable output parameter `OosAct = 1`. You can interconnect the output parameter `OosAct` of the channel block with the input parameter `OosLi` of a technologic block.

Use the `Feature` bit Reaction to the out of service mode (Page 188) to specify, in case the input parameter is `OosLi = 1`, if:

- there is a switchover to the "out of service" mode and the symbol for the "In progress" (see table) status is displayed. You can change to manual mode at any time.
- Only the "In progress" display (see table) in the block icon and in the faceplate of the assigned technologic block is made.

| Display | Meaning |
|---|---|
|  | In progress |

#### Function sequence in the APL

- The OS operator issues the release for maintenance (`MS_RelOp = 1`) in the technologic block's parameter view.
- The technologic block then sets the `MS_Release = 1` output parameter.
- The channel block's `MS_Rel` input is now also 1.

- The channel block signals the release for maintenance to the diagnostics block via the `DXCHG` parameter.
- The release for maintenance is only signaled to the Maintenance Station once all 0 bits of the parameter `DXCHG_XX` are set on the diagnostics block.

- The channel block determines the "in progress" state of the Maintenance Station using the `MS` input parameter and makes this information available at the `OosAct` output parameter.
- On the technologic block, the "working" state is displayed at input parameter `OosLi` and forwarded for display to the faceplate.

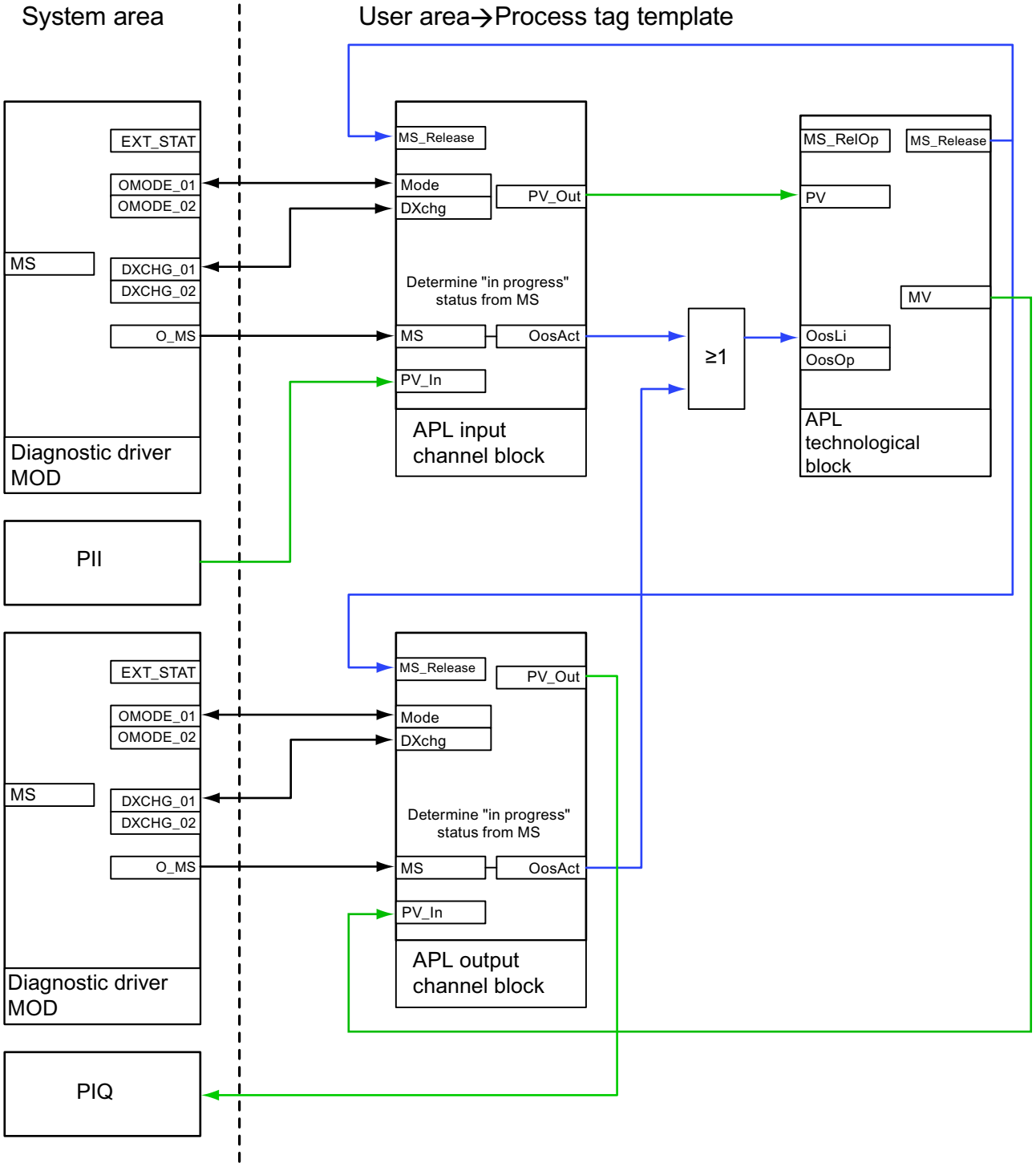



Figure 2-6 Release for maintenance

**Key to diagram:**

| | |
|---|---|
| PII | Process image of inputs |
| PIQ | Process image of outputs |
| Black lines | Automatic system connections |
| Green lines | Process value connections made by the planner |
| Blue lines | Connections for release for maintenance made by the planner |

**Note**

For additional information on the topic of maintenance please refer to PCS 7 OS process management.

### 2.1.1.13 SIMATIC BATCH functionality

#### SIMATIC BATCH functionality

Some blocks have an interface to SIMATIC BATCH. You use them when you connect `BatchEn`, `BatchID`, `BatchName`, `StepNo` and `Occupied` I/Os to the corresponding SIMATIC BATCH blocks. A new `BatchName` becomes effective if the `BatchID` is changed. Refer to the SIMATIC BATCH documentation.

Please refer to the descriptions of the individual blocks for information about whether a block supports the SIMATIC BATCH functionality.

### 2.1.1.14 Flutter suppression for channel blocks

#### Flutter suppression

The time-controlled "Flutter suppression" function is used to delay the outgoing of a message by a configurable period.

Flutter suppression is used for

- OB82 events - diagnostic messages
- OB83 events - fault

used.

The flutter time is entered at the channel block at the `FlutTmIn` parameter. The high byte of the `DataXchg` parameter of the channel blocks contains the flutter time.

Flutter suppression comes into effect when `FlutEN` = 1 and `FlutTmIn` > 0 is set at the channel block.

There is only one flutter message per module. The delay times and fault messages are channel-specific. The fault messages are extended by at least the delay time. Flutter occurs when the status of fault messages changes from "Outgoing" back to "Incoming" within the delay time.

The last fluttering channel and its set delay time deactivates the flutter message.

The following channel blocks have this function:

FbAnIn - Analog input channel block for field devices (Page 2466)

FbAnTot - Analog totalizer channel block for field devices (Page 2487)

FbAnOu - Analog output channel block for field devices (Page 2476)

FbDiIn - Digital input channel block for field devices (Page 2495)

FbDiOu - Digital output channel block for field devices (Page 2505)

Pcs7AnIn - Analog input channel block (Page 2564)

Pcs7AnOu - Analog output channel block (Page 2577)

Pcs7DiIn - Digital input channel block (Page 2596)

Pcs7DiIT - Digital input channel block with time stamp (Page 2605)

Pcs7DiOu - Digital output channel block (Page 2616)

Pcs7Cnt1 - Controlling and reading FM 350 modules (Page 2625)

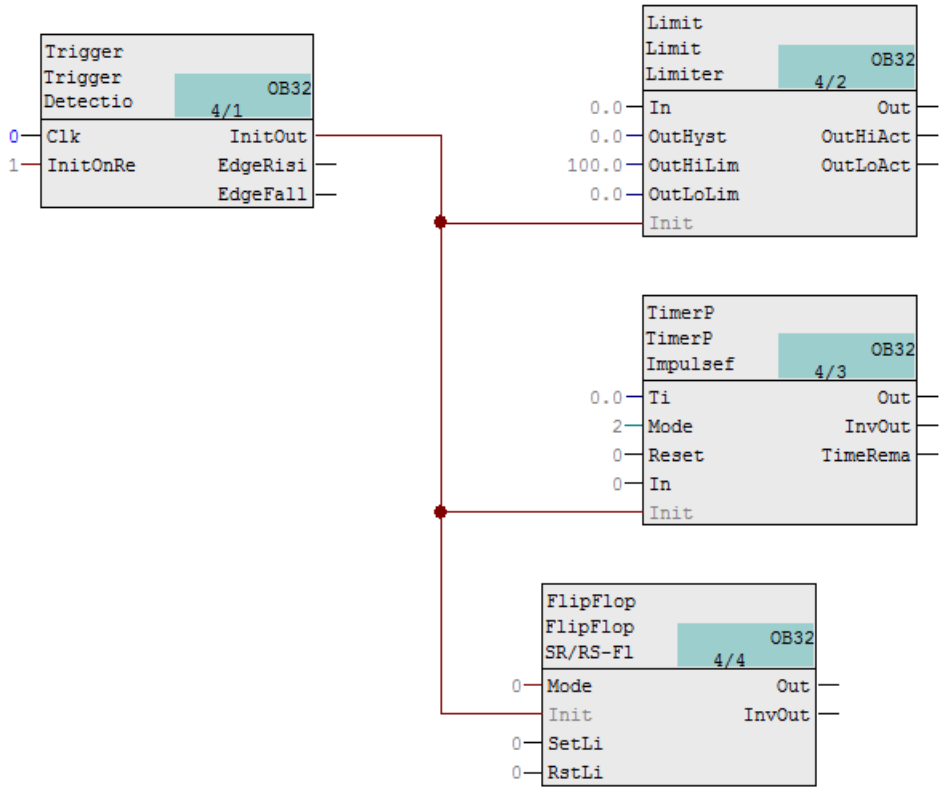
Pcs7Cnt2 - Control and read an 8-DI_NAMUR module of the ET 200iSP (Page 2636)

Pcs7Cnt3 - Control and read the 1 COUNT 24V/100kHz module for count mode (Page 2646)

### 2.1.1.15 Startup characteristics over Trigger block

#### Startup characteristics over Trigger block

With the function block Trigger, you can influence the startup characteristics of the blocks Limit, TimerP, FlipFlop, and Trigger. Connect the output parameter `InitOut` of the Trigger block with the input parameter `Init` of the block Limit, TimerP, or FlipFlop to initialize these blocks. In the second cycle of OB30 to OB38, the output parameter `InitOut` of the Trigger block will be reset after a startup.




### 2.1.1.16 Parameters that cannot be connected

This section provides information on parameters that cannot be connected by user via CFC connection. (for example: `StartValue` at the Pcs7DiOu).

the following parameters that are not linkable:

| Parameters | Applicable for all the blocks | TimeTrig |
|---|---|---|
| `SampleTime` | x | x |
| `MsgEvId(x)` | x | x |
| All inputs beginning with Old.<br>For example: In CntOhSc: `OldMinutes`, `OldDays`, and so on. | x | x |

| Parameters | Applicable for all the blocks | TimeTrig |
|---|---|---|
| J_Mini | x | x |
| OptimOcc | x | x |
| InPerTrigOn | - | x |
| InSglTrigOn | - | x |
| InSglTrigDT | - | x |

## 2.1.2 Operating modes of the blocks

### 2.1.2.1 Overview of the modes

#### Overview of the individual modes

The available operating modes are assigned to the block families:

- Motors, valves and dosers
- Controllers
- Blocks without "Manual" and "Automatic" modes

You can find an overview below. Click on one of the operating modes to go directly to the relevant detailed description.

You can find a state graph for the operating modes (Page 90) at the end of this section.

#### Operating modes for motors, valves and dosers

The following operating modes are available:

1. Local mode (Page 86)
2. Automatic mode (Page 82)
3. Manual mode (Page 82)
4. Out of service (Page 77)

The mode with the lowest number in the list above has the highest priority. "Manual" and "Automatic" modes have the same priority. General information on the individual modes is available in the following sections. The sections also include block-specific information, for example, non-standard parameter assignment. Refer to the description and function of the relevant blocks.

#### Operating modes for controllers

The following operating modes are available:

1. Automatic mode (Page 79)
2. Manual mode (Page 79)

3. Program mode for controllers (Page 85)
4. Out of service (Page 77)

The mode with the lowest number has highest priority. "Manual" and "Automatic" modes have the same priority. General information on the individual modes is available in the following sections. The sections also include block-specific information, for example, non-standard parameter assignment. Refer to the description and function of the relevant blocks.

### Operating modes for blocks without "Manual" and "Automatic" operation

The following operating modes are available:

1. On (Page 77)
2. Out of service (Page 77)

The mode with the lowest number has highest priority. General information on the individual modes is available in the following sections. The sections also include block-specific information, for example, non-standard parameter assignment. Refer to the description and function of the relevant blocks.

**Note**

Note that the operating modes are realized differently in the individual block families.

## 2.1.2.2 On

### "On" operating mode

The "On" operating mode tells you that the block algorithm is being processed (output parameter `OnAct = 1`). This operating mode is only available for blocks that have faceplates but not the following operating modes:

- Manual mode or
- Automatic mode or
- Local mode

The "On" mode can only be activated via a control on the faceplate (input parameter `OnOp = 1`). The block must be in the "Out of service" operating mode for this to be possible.

## 2.1.2.3 Out of service

### Using the "Out of service" operating mode

The "Out of service" operating mode is available to all blocks that have an operating mode switchover and a direct connection to the process (with a connection to a process tag, for example).

The "Out of service" operating mode is intended for purposes of maintenance and servicing (replacing the device, for example). All of the block's functions are disabled. No incoming or

outgoing messages are generated. The only function still possible is an operating mode switchover.

All outputs for motors and valves are set to the neutral position in this operating mode.

The timer for the function "Restart disable after the shutoff of motor (Page 1762)" is reset in this mode.

**Note**

**Notes on VlvMotL and VlvPosL**

If the neutral position of is "Closed" or "Open", the corresponding control outputs "Closed" or "Open" are set. The control outputs are reset after reaching the neutral position or with an active torque signal. The "Seal valve" function (`Feature.Bit8`) is a component.

**Note**

**Note on VlvMotL**

If the timer of the "Restart disable after the shutoff of motor (Page 1762)" function is running and the block is not in the neutral position, it is not possible to switch to "Out of service" mode.

**Note**

For the `RbkOut.ST` parameter, the status is set to "Out of service (OOS)" in the moment of change and not in every cycle of OOS.

For controllers, the neutral position manipulated variable (high or low manual limit of the manipulated variable) is only used if the `Feature` bit Neutral position manipulated variable takes effect at startup (Page 176) is active. Otherwise the manipulated variable remains at the latest value like all the other output parameters.

See also the section Neutral position for motors, valves and controllers (Page 54) for more on this.

The last value available is output permanently for all other blocks.

### Requirement for the "Out of service" mode

Prerequisite for switching to this operating mode is that the block is in "Manual mode" or "On" mode.

### Activating the "Out of service" operating mode using the faceplate

The "Out of service" operating mode can only be switched on by using the faceplate when it is in the default block view (`OosOp = 1` parameter) and even then, only if `ModLiOp = 0`.

To switch the operating mode using the faceplate, refer to the descriptions relating to the standard view of the individual blocks.

### Switching on the "Out of service" operating mode by using the interconnection

The "Out of service" operating mode is switched on by using the configurable parameter `OosLi = 1`. This is only possible if the block is in manual mode or "On" mode and the `Feature` bit Reaction to the out of service mode (Page 188) was set to 1.

Regardless of operating mode, the parameter view of the faceplate will always display the status of the parameter `OosLi =1` with the symbol for the status "In progress" (see table) next to the Maintenance enable button.

| Display | Meaning |
|---|---|
|  | In progress |

See also the section Release for maintenance (Page 70) for more on this.

### Exiting the "Out of service" operating mode

From this operating mode, a block can only be switched by an operator action at the faceplate into the following operating modes:

- "On"
- "Manual mode"

## 2.1.2.4 Manual and automatic mode for control blocks

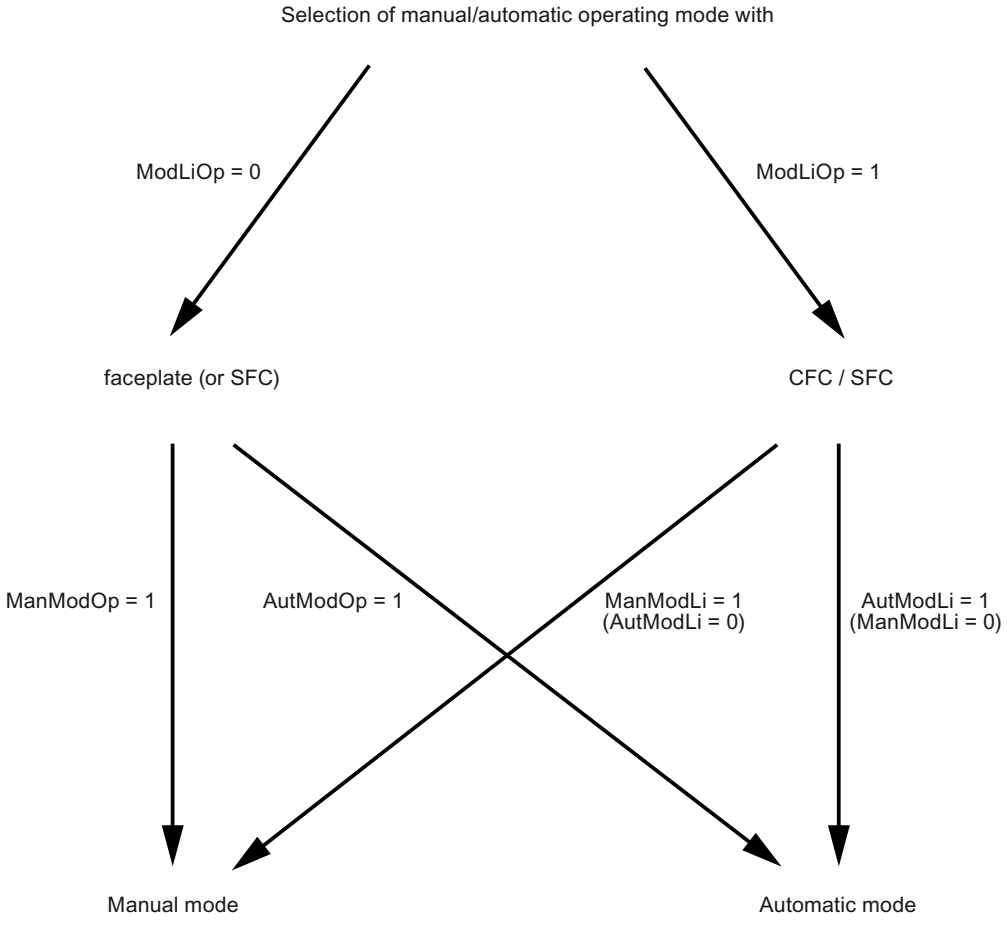
### "Manual" and "Automatic" modes for controller blocks

In "manual mode", the control settings for the device are made manually by the operator. The operator decides how to change the block's manipulated variable (output signal). The manipulated variable can be analog or binary.

In "automatic mode", the control settings for the controller are made automatically as calculated by the block algorithm.

### Changing between operating modes

The switchover between manual and automatic modes takes place as shown in the following schematic:




**Switchover initiated in the faceplate** (`ModLiOp` = 0)**:** The switchover between operating modes is carried out in the standard view of the faceplate. In the function block, the parameters `ManModOp` for "manual mode" and `AutModOp` for "automatic mode" are used.

If both signals (`ManModOp =1`, `AutModOp = 1` ) are set, `ManModOp = 1` has priority.

**Switchover per interconnection (CFC or SFC instance)** (`ModLiOp = 1`)**:** The switchover between the operating modes is carried out with an interconnection on the function block. The parameters `ManModLi` for "manual mode" and `AutModLi` for "automatic mode" are used in pushbutton operation. In switching mode (requirement: `Feature.Bit4` = 1, see Setting switch or button mode (Page 177)) connection `AutModLi` is used exclusively.

If both signals (ManModLi =1, AutModLi = 1), the previous operating mode is retained and the ErrorNum output is assigned with the error number 51.

**Note**

You can access the variable parameters AutModOp and ManModOp from a normal SFC (in contrast to the instance of an SFC type). The SFC can thus change the operating mode without revoking the access rights of the operator (i.e. without setting ModLiOp = 1).

### Switchover from automatic mode to manual mode

When changing over from "automatic mode" to "manual mode", the last valid control settings (**M**anipiulated **V**alue MV)for the controller set in "automatic mode" remain valid until you change the control settings manually.

### Switchover from automatic mode to manual mode

The switchover from manual to automatic mode can take place with or without the internal setpoint tracking the process value. You specify this behavior on the SP_TrkPV I/O, which can also be operated from the faceplate in the parameter view (Option "SP = PV"). For the blocks PIDConL and PIDStepL you can also change the behavior for the switchover via the parameter Feature bit Disabling bumpless switchover to automatic mode for controllers (Page 183):

- **Switchover with internal setpoint tracking process variable**(SP_TrkPV = 1) means that in "Manual" mode the setpoint (SP) tracks the process value (PV) (bumpless switchover). After switching back to "Automatic" mode, the manipulated variable remains constant until the setpoint value (SP) is changed or the process value (PV) changes.
- **Switchover without internal setpoint tracking process variable** (SP_TrkPV = 0) means that the block immediately recalculates the value of the manipulated variable based on the setpoint and process value (PV) when the mode is changed. The Feature parameter is used to choose between the two variants:
  - **Switchover without P step** (standard setting, Feature bit = 0):
    During switchover, the I action of the controller is set in such a way that the switchover is carried out without a P step (virtually bumpless referring to the manipulated variable). A control deviation is only regulated via the I action.
  - **Switchover with P step** (Feature bit = 1):
    During switchover, the I action of the controller is set in such a way that the switchover is carried out with a P step (not bumpless referring to the manipulated variable). A control deviation is regulated via the P and the I action.

    **Note**

    **Points to note about switchovers with a P step change:**

    - The P action must be active for the setting "Switchover with P step" (PropSel = 1)
    - If the P action is in the feedback (PropFacSP = 0), the "Switchover with P step" setting has no effect.
    - If the switchover function with the internal setpoint tracking the process variable is active (SP_TrkPV = 1), the "Switchover with P step" setting has no effect.

### Reaction of signals when operating mode is changed

Using the `Feature` bit Resetting the commands for changing the mode (Page 172), you can choose whether the block automatically resets the signal for changing the operating mode.

### Switch on program mode

A few controller blocks allow you to operate in program mode. Refer to the relevant sections for the controller blocks to learn whether a control block allows program mode.

Also refer to the section Program mode for controllers (Page 85) for information on program mode.

## 2.1.2.5 Manual and automatic mode for motors, valves and dosers

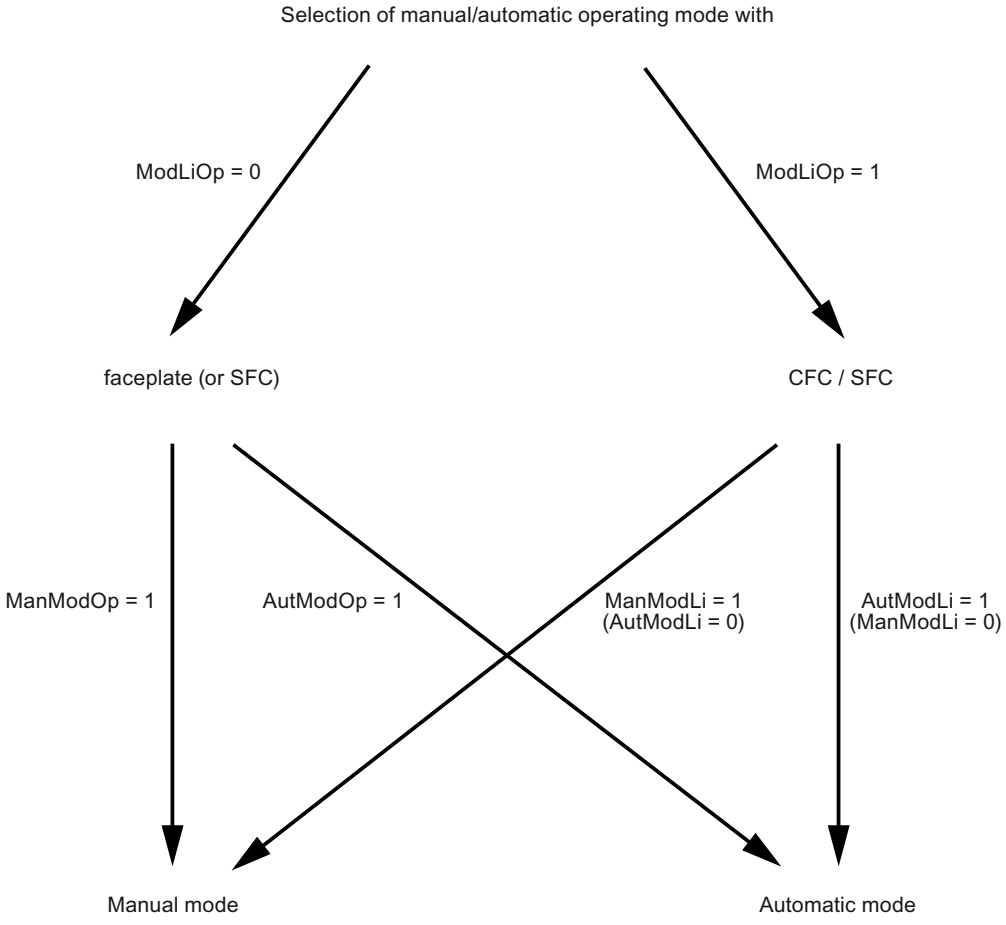
### Manual and automatic mode for motors, valves and dosers

In "manual mode", the control settings for the device are made manually by the operator. The operator decides how to change the block's manipulated variable (output signal). The manipulated variable can be analog or binary in accordance with the function block.

In "automatic mode", the control settings for the device are made by the block algorithm via interconnected inputs or inputs controlled by SFC.

## Changing between operating modes

The switchover between "manual and automatic mode" takes place as shown in the following schematic:




**Note**

The two selections (manual and automatic) cannot both be set to "1" in switching mode.

**Switchover using faceplates** (`ModLiOp = 0`)**:** The switchover between operating modes is carried out in the standard view of the faceplate. In the function block, the parameters `ManModOp` for "manual mode" and `AutModOp` for "automatic mode" are used.

**Switchover per interconnection (CFC or SFC instance)** (`ModLiOp = 1`)**:** The switchover between the operating modes is carried out with an interconnection on the function block. The parameters `ManModLi` for "manual mode" and `AutModLi` for "automatic mode" are used in

pushbutton operation. In switching mode (requirement: `Feature.Bit4` = 1, see Setting switch or button mode (Page 177)) connection `AutModLi` is used exclusively.

**Note**

The `Feature.Bit4` is available only for the "large" blocks.

**Note**

You can access the variable parameters `AutModOp` and `ManModOp` from a normal SFC (in contrast to the instance of an SFC type). The SFC can thus change the operating mode without revoking the access rights of the operator (i.e. without setting `ModLiOp = 1`).

### Switchover from automatic mode to manual mode

When changing over from "automatic mode" to "manual mode", the last valid control settings for the block set in "automatic mode" remain valid until you change the control settings manually.

**Note**

For VlvAnL, the bumplessness of the switchover depends on the parameter `MV_TrkExt`.

### Switchover from manual to automatic mode

You can set the following options for changing over from "manual mode" to "automatic mode" using the `Feature` bit Bumpless switchover to automatic mode (Page 184). Refer to the I/O descriptions for the relevant block.

- A switchover from manual to automatic mode is possible at any time (standard setting, `Feature` bit = 0). The control settings for the automatic mode become effective immediately.
- Switchover from manual to automatic mode is only possible if the control settings for the manual and automatic modes match (Bumpless switchover), (`Feature` bit = 1). An error message is output if they do not match. In this case, you will need to adapt the control settings in "manual mode" to the control settings in "automatic mode".

  **Note**

  The "Bumpless switchover to automatic mode" function is supported only by "large" blocks.

### Reaction of signals when operating mode is changed

Using the `Feature` bit Resetting the commands for changing the mode (Page 172), you can choose whether the block automatically resets the signal for changing the operating mode.

### Resetting the commands for the control settings

With the `Feature` bit Enabling resetting of commands for the control settings (Page 172), you select how the block handles commands for the control settings (for example motor on) via the interconnected input parameters.

## 2.1.2.6 Program mode for controllers

### Program mode for controllers - interface for higher-level control functions

The interface for primary controller functions (external Advanced Control software package) provides primary controller functions, which run on an external PC as an OPC client, the option of using the control from the controller function block and specifying the setpoint or manipulated variable from a remote location. This procedure is called program mode.

You can use the `Feature` bit Enabling program mode (Page 170) to specify whether or not the controller block is intended for program mode.

Program mode requires an enable signal (input parameter `AdvCoEn = 1`) from a central control block. If this enable signal goes from 1 to 0, for example, due to errors in the OPC communication, the controller block switches to manual mode if it is in program mode with manipulated variable specification or to automatic mode if it is in program mode with setpoint specification.

You activate program mode in the standard view of the controller faceplate. In addition to switching from manual to automatic mode, you are also given the option of using program mode as the operating mode. You exit program mode by operator input or by switching back into manual or automatic mode.

A 0-1 edge transition of the interconnectable input parameter `AdvCoMstrOn` activates program mode depending on the conditions described below. You can use this to put an entire group of downstream controller blocks into program mode at the same time from a central control block. Both the input parameter `AdvCoOn` and the interconnectable input parameter `AdvCoMstrOn` can be used at the same time, since the parameter `AdvCoMstrOn` only reacts to edges of the binary signal.

Program mode is deactivated with a 1 - 0 edge transition.

**Note**

The output parameters `ManAct` or `AutAct` display the mode to which the controller changes with a 1 - 0 edge transition.

The output parameter `AdvCoRdy = 1` indicates if the PID controller is ready to switch to program mode. At a central control block, you can use an AND operation for all `AdvCoRdy` signals of the downstream controllers to enable central switchover.

The output parameter `AdvCoAct = 1` indicates of the block is in program mode.

**Selecting the type of program mode**

There are two types of program mode:

- Program mode driven by setpoint (in automatic mode only)
- Program mode driven by manipulated variable (in manual mode only, not for step controllers without position feedback)

**Program mode with setpoint**: If you set the input parameter `AdvCoModSP = 1`, the analog value provided by the OPC client (`AdvCoMV`) is used as an external setpoint for the controller. The controller and faceplate otherwise react as they do with automatic mode and an external setpoint. Refer to section Setpoint specification - internal/external (Page 136) for more about this.

Requirements for program mode with setpoint:

- `AdvCoModSP = 1`,
- `AdvCoEn = 1`,
- Controller has to be in automatic mode
- The setpoint can be set to internal as well as external.

**Program mode with manipulated variable**: If you set the input parameter `AdvCoModSP = 0`, the analog value provided by the OPC client (`AdvCoMV`) is used as an external manipulated variable for the controller. The algorithm of the PID controller is bypassed. The controller and faceplate otherwise react as they do with tracking (`MV_TrkOn = 1`). Refer to section Tracking and limiting a manipulated variable (Page 222) for more about this.

Requirements for program mode with manipulated variable:

- `AdvCoModSP = 0`,
- `AdvCoEn = 1`,
- Controller is preferably in manual mode, but it can also be in automatic mode.
- The setpoint can be set to internal as well as external.

**Note**

Program mode with manipulated variable is not available for step controllers without position feedback (available in PIDStepL, FmCont and FmTemp). `ErrorNum = 50` is output on the controller block and the controller cannot switch into program mode `(AdvCoAct=0)`.

### 2.1.2.7 Local mode

#### Areas of application for local mode

This operating mode is used for motors, valves and dosing units. The control settings are made directly or via a control station that is located "locally". In addition, you can set different control strategies with the parameter `LocalSetting`.

With `LocalSetting = 0`, you prevent a change to "local mode".

**Note**

**Differences between "Large" and "Small" blocks**

The operating mode described here is valid for "Large" blocks. For "Small" blocks, `LocalSetting` can be configured only on a limited basis. For more information, refer to the respective description for the operating modes of the blocks.

### Changing to local mode

Changing to local mode is only possible from the manual and automatic operating modes. The change to this mode is initiated by:

- An operation on the faceplate (input parameter `LocalOp = 1`, valid if `LocalSetting = 3` or `LocalSetting = 4` and `ModLiOp = 0`) or
- The interconnected input parameter (`LocalLi = 1`, valid if `LocalSetting = 1` or `LocalSetting = 2`).

### Exiting local mode

You leave local mode using:

- An operation on the faceplate (`LocalSetting = 3` or `LocalSetting = 4` and `ModLiOp = 0`) or
- the interconnected input parameter (`LocalSetting = 1` or `LocalSetting = 2`). In order to exit local mode via the interconnected input parameter, you can configure various reactions using a `Feature` bit Exiting local mode (Page 189).

### Operator input in "local mode" using a faceplate

You are not permitted to functionally operate the block in local mode. You can only use the faceplate to exit local mode if you have also activated "local mode" using the faceplate. The rules you specified for exiting "local mode" apply here.

### Input in "local mode" via interconnected inputs

In "local mode", the way the block functions is influenced via interconnected input parameters according to the settings of the `LocalSetting` parameter. You have the following options:

- `LocalSetting = 1` and `LocalSetting = 3`
  - The control settings for the block are adjusted (tracking) via an interconnected input parameter. The interconnected input parameter includes the control signal for the local operator station on the system.
  - The runtime monitoring of the block is effective in accordance with your configuration.
  - The interlocking functions of the block are activated in accordance with input parameter `BypProt = 0` or deactivated (`BypProt = 1`).

    **Note**

    The block VlvAnL does not support `LocalSetting` = 1/3.

- `LocalSetting = 2` and `LocalSetting = 4`
  - The control settings for the block are made based on internal adjustment of the feedback value.
  - Runtime monitoring of the block is active if more than one available feedback signals are set (discrepancy).
    An exception is VlvMotL:

    **Note**

    **VlvMotL**

    The motor and valve feedback signals are monitored if the motor feedback signlas exist and are connected with `FbkOpening` and `FbkClosing` (`Feature.Bit2` = 0). The motor feedback signals are only monitored in the end positions of the valve and in discrepancy. For example, if the valve is in end position `FbkOpen` =1 and the motor feedback `FbkOpening` =1 is pending, an error message is generated upon expiration of the monitoring time.

    **VlvAnL:** The auxiliary valve is controlled via internal tracking of the feedback signals `FbkAuxVCloseOut` and `FbkAuxVOpenOut`. The control of the main valve via the feedback value `Rbk` is not affected by this.

The texts for labeling the command buttons in the faceplates of the motor and valve blocks can now be assigned for each specific instance.
The configuration of the texts is performed with the "Text 1" property of the respective control inputs of the motor and valve blocks in the CFC.

If no instance-specific text is configured, the previous default texts are used and displayed in the faceplate.

The following table shows the assignment of the command buttons to the corresponding block input:

The interlock functions of the block are deactivated.

## Overview of behavior in local mode

| **LocalSetting=** | **0** | **1** | **2** | **3** | **4** | **5 (VlvS only)** |
|---|---|---|---|---|---|---|
| Switch on operating mode | Cannot be set | CFC/SFC | CFC/SFC | Faceplate | Faceplate | CFC |
| Changing the operating mode:<br>Local mode/to manual mode only<br>(`Feature` = 0) | - | CFC/SFC | CFC/SFC | - | - | CFC/SFC |
| Changing the operating mode:<br>Local mode/previous mode (`Feature` = 1) | - | CFC/SFC | CFC/SFC | - | - | CFC/SFC |
| Operating in the faceplate | - | Resetting of Errors/Protection Via Faceplate, Resetting of Rapid Stop | Rapid stop and resetting of rapid stop (only for "Large" blocks). Resetting of Errors/Protection Via Faceplate, | Only switching of operating mode, rapid stop, internal/ external setpoint switchover and resetting of rapid stop | Only switching of operating mode, rapid stop, and resetting of rapid stop | - |
| Executing local commands | - | Yes | No | Yes | No | No |
| Reaction of the block | - | Monitoring the feedbacks | Tracking of feedback, monitoring feedback during rapid stop, external errors, motor protection or discrepancy | Monitoring the feedbacks | Tracking of feedback, monitoring feedback during rapid stop, external errors, motor protection or discrepancy | Monitoring the feedbacks |
| Interlock activated | - | Yes:<br>(`BypProt = 0`)<br>No:<br>(`BypProt = 1`) | only at output `LockAct` with `Feature.Bit27` = 1 and `BypProt = 0` | Yes:<br>(`BypProt = 0`)<br>No:<br>(`BypProt = 1`) | only at output `LockAct` with `Feature.Bit27` = 1 and `BypProt = 0` | only at output `LockAct` with `Feature.Bit27` = 1 and `BypProt = 0` |

### 2.1.2.8 State graph of the operating modes

## State graph of the operating modes

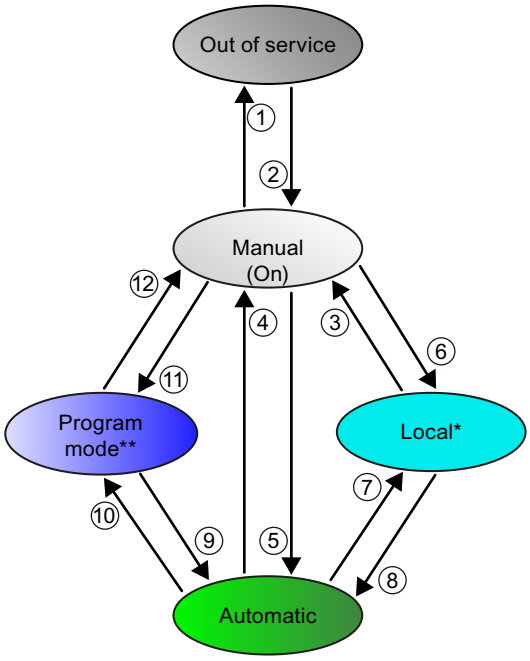



Figure 2-7 State graph of the operating modes

* This operating mode is used for motors, valves, and dosing units.

** This operating mode is used for controllers only.

| Number in graphic (top) | Condition for status change |
|---|---|
| (1) | **Manual (on) → Out of service**<br>• Via faceplate `(OosOp = 1)` if `ModLiOp = 0` or<br>• Via edge transition 0 → 1 of `OosLi` if `Feature` bit Reaction to the out of service mode (Page 188)= 1 |
| (2) | **Out of service → Manual (on)**<br>• Via faceplate `(ManModOp = 1)` |
| (3) | **Local mode → Manual**<br>• Via faceplate `(ManModOp = 1)` if `ModLiOp = 0` and `LocalSetting = 3` or `LocalSetting = 4` or<br>• Via `LocalLi = 0` if `LocalSetting = 1`, `LocalSetting = 2` or `LocalSetting = 5`. See section Exiting local mode (Page 189) for more conditions. |

| Number in graphic (top) | Condition for status change |
|---|---|
| (4) | **Automatic → Manual**<br>• Via faceplate (`ManModOp = 1`) if `ModLiOp = 0` or<br>• Via `ManModLi = 1` if `ModLiOp = 1` and `Feature` bit Setting switch or button mode (Page 177)`= 0` or<br>• Via `AutModLi = 0` if `ModLiOp = 1` and `Feature` bit Setting switch or button mode (Page 177)`= 1` |
| (5) | **Manual → Automatic**<br>• Via faceplate (`AutModOp = 1`) if `ModLiOp = 0` or<br>• Via `AutModLi = 1` if `ModLiOp = 1` |
| (6) | **Manual → Local mode**<br>• Via faceplate `(LocalOp = 1)` if `ModLiOp = 0` and `LocalSetting = 3` or `LocalSetting = 4` or<br>• Via `LocalLi = 1` if `LocalSetting = 1`, `LocalSetting = 2` or `LocalSetting = 5` |
| (7) | **Automatic → Local mode**<br>• Via faceplate `(LocalOp = 1)` if `ModLiOp = 0` and `LocalSetting = 3` or `LocalSetting = 4` or<br>• Via `LocalLi = 1` if `LocalSetting = 1`, `LocalSetting = 2` or `LocalSetting = 5` |
| (8) | **Local mode → Automatic**<br>• Via faceplate `(AutModOp = 1)` if `ModLiOp = 0` and `LocalSetting = 3` or `LocalSetting = 4` or<br>• Via `LocalLi = 0` if `LocalSetting = 1`, `LocalSetting = 2` or `LocalSetting = 5`. See section Exiting local mode (Page 189) for more conditions. |
| (9) | **Program mode → Automatic**<br>• Via faceplate (`AutModOp = 1`) if `ModLiOp = 0` or<br>• Via `AutModLi = 1` if `ModLiOp = 1` or<br>• Via edge transition 1 → 0 of `AdvCoMstrOn` if automatic is set before program mode. |
| (10) | **Automatic → Program mode**<br>Requirement for switchover in program mode: `AdvCoEn = 1`<br>• Via faceplate (`AdvCoOn= 1`) if `ModLiOp = 0` or<br>• Via `AdvCoMstrOn = 1` |
| (11) | **Manual → Program mode**<br>Requirement for switchover from manual to program mode: `AdvCoEn = 1` and `AdvCoModSP = 0`<br>• Via faceplate (`AdvCoOn= 1`) if `ModLiOp = 0` or<br>• Via `AdvCoMstrOn = 1` |
| (12) | **Program mode → Manual**<br>• Via faceplate (`ManModOp = 1`) if `ModLiOp = 0` or<br>• Edge transition 1 → 0 of `AdvCoMstrOn` if manual is set before program mode. |

## 2.1.3 Monitoring functions

### 2.1.3.1 Monitoring functions in the Advanced Process Library

#### Monitoring functions in the Advanced Process Library

This and the following chapters encompass the standard monitoring functions in the Advanced Process Library. The monitoring functions include:

- Limit value monitoring
- Feedback monitoring
- Motor protection

Some of the configured time values (e.g. `MonTiStatic`, `MonTiDynamic`) are limited at the low end to the sampling time by the block algorithm and written back to the block input. "Reset Program" (after a "Download Entire Program" for example) writes the parameter values changed in this way to the offline data storage system.

For further and detailed information, refer to the following chapters. For the block-specific monitoring functions, also refer to the description of the particular block.

### 2.1.3.2 Group display for limit monitoring, CSF and ExtMsgx

#### Group display for limit monitoring, CSF and ExtMsgx

The SumMsgAct output parameter assembles the following signals of a block and makes them available to you:

- Limit monitoring of the process value
- Limit monitoring of the count value
- Limit monitoring of the feedback
- Limit monitoring of setpoint difference, manipulated variable difference and error signal
- External control system fault (CSF)
- Freely selectable messages ExtMsg1..4

**Note**

The signal status of the individual signals is not taken into consideration for forming the group error.

### 2.1.3.3 Limit monitoring

#### Limit monitoring of the process value

You can monitor the process value to the following high and low alarm, warning and tolerance limits:

- `PV_AH_Lim`: Limit for high alarm
- `PV_AL_Lim`: Limit for low alarm
- `PV_WH_Lim`: Limit for high warning
- `PV_WL_Lim`: Limit for low warning
- `PV_TH_Lim`: Limit for the high tolerance
- `PV_TL_Lim`: Limit for the low tolerance

**Note**

The message classes Alarm, Warning and Tolerance are not valid for user-configured message classes. Please take into consideration the validity of terms for user-configured message classes (Page 45).

**Note**

**Special note for "Small" blocks**

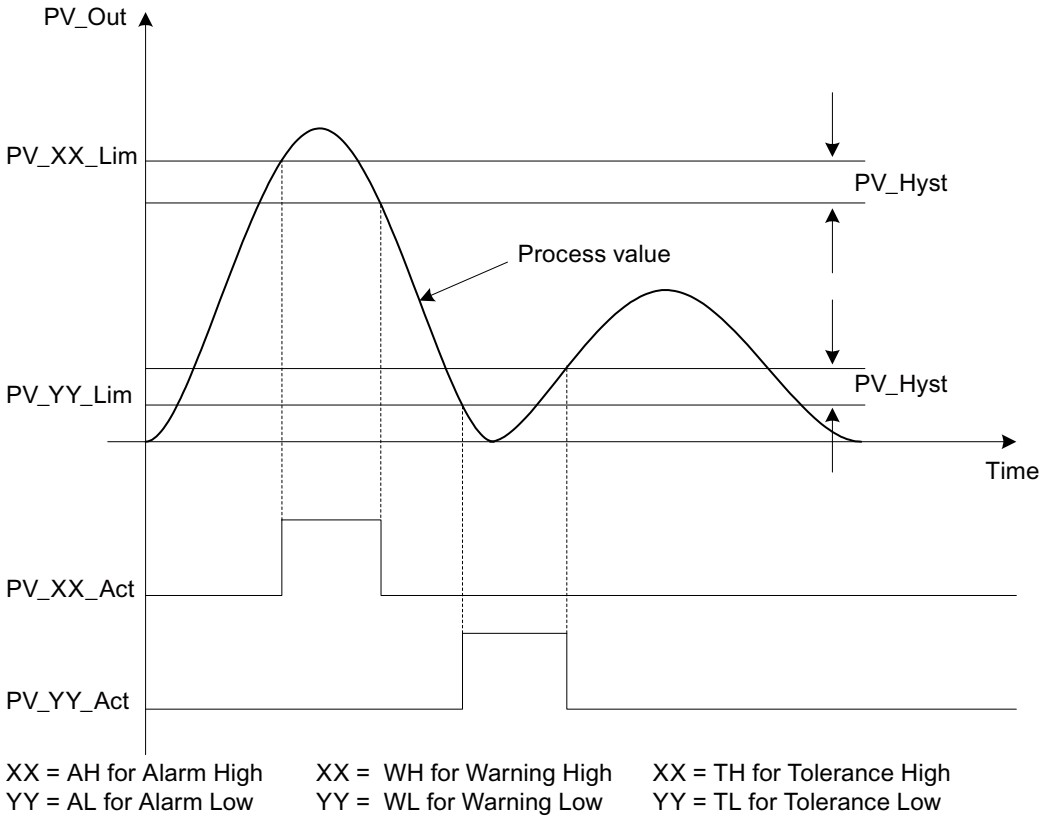
"Small" blocks provide only the monitoring for alarms and warnings.

#### Result of the limit monitoring

The result of limit monitoring is made available at the interconnectable output parameters:

- `PV_AH_Act = 1`: Limit for high alarm reached or exceeded
- `PV_AL_Act = 1`: Limit for low alarm reached or undershot
- `PV_WH_Act = 1`: Limit for high warning reached or exceeded
- `PV_WL_Act = 1`: Limit for low warning reached or undershot
- `PV_TH_Act = 1`: Limit for high tolerance reached or exceeded
- `PV_TL_Act = 1`: Limit for low tolerance reached or undershot

made available (see figure). The `SumMsgAct = 1` output parameter is also set when at least one limit value is reached or violated.




You can use `Feature.Bit29` Signaling limit violation (Page 181) to determine whether the output parameter triggers limit monitoring with the value "0" or "1".

You can use `Feature.Bit28` Disabling operating points (Page 156) to disable limit monitoring when message suppression is enabled (`MsgLock = 1`).

### Activating limit monitoring

Monitoring is always enabled using the input parameters:

- `PV_AH_En = 1`: Monitoring of the high alarm limits
- `PV_AL_En = 1`: Monitoring of the low alarm limits
- `PV_WH_En = 1`: Monitoring of the high warning limits
- `PV_WL_En = 1`: Monitoring of the low warning limits
- `PV_TH_En = 1`: Monitoring of the high tolerance limits
- `PV_TL_En = 1`: Monitoring of the low tolerance limits

**Predefinition:** When the block is installed, monitoring of the tolerance limits is disabled, meaning that the parameters are configured with 0. To activate monitoring, assign 1 to these parameters.

All other monitoring functions are enabled.

### Message suppression

| | |
|---|---|
|  | Symbol for message suppression |

The corresponding message is suppressed using the parameters:

- `PV_AH_MsgEn = 0`: Alarm (high) messages are suppressed
- `PV_AL_MsgEn = 0`: Alarm (low) messages are suppressed
- `PV_WH_MsgEn = 0`: Warning (high) messages are suppressed
- `PV_WL_MsgEn = 0`: Warning (low) messages are suppressed
- `PV_TH_MsgEn = 0`: Tolerance (high) messages are suppressed
- `PV_TL_MsgEn = 0`: Tolerance (low) messages are suppressed

The output of messages is not suppressed when the block is installed (all `xx_MsgEn` parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

### Hysteresis

You can specify a hysteresis (`PV_Hyst`) for the limits, for example, to suppress signal flutter. Refer to the Limit monitoring with hysteresis (Page 104) chapter for more on this.

### Alarm delays

You can set alarm delays for incoming and outgoing alarms, warnings and tolerances. Refer to the Area of application of the alarm delays (Page 225) chapter for more on this.

### Operating in the faceplate

You can also change the limits and the hysteresis using the faceplate. Refer to the Limit operation and display in the faceplate (Page 355) chapter for more on this.

### See also

Two time values per limit pair (Page 227)

Two time values for each individual limit (Page 228)

User-configured message classes (Page 45)

### Limit monitoring of the count value

You can monitor the count value to the following high and low alarm, warning and tolerance limits:

- `OutAH_Lim`: Limit for high alarm
- `OutAL_Lim`: Limit for low alarm
- `OutWH_Lim`: Limit for high warning
- `OutWL_Lim`: Limit for low warning
- `OutTH_Lim`: Limit for the high tolerance
- `OutTL_Lim`: Limit for the low tolerance

**Note**

The message classes Alarm, Warning and Tolerance are not valid for user-configured message classes. Please take into consideration the validity of terms for user-configured message classes (Page 45).

**Note**

**Monitoring the limits**

The limits monitored depend on the direction of counting:

- In Mode 1 (summing up or integrating), the high limits are monitored:
  - OutAH_Lim
  - OutWH_Lim
  - OutTH_Lim
- In Mode 2 (summing down or integrating), the low limits are monitored:
  - OutAL_Lim
  - OutWL_Lim
  - OutTL_Lim

### Example for limit monitoring with a counter

Monitoring of the high limit is performed only if the counter is running in "up" direction.

If you count "up" but with negative values, for example, from 100 down, you need to adjust the high limit accordingly (for example, the high limit could be -15).

This behavior applies to the blocks CountScL, CountOh and TotalL.

### Result of the limit monitoring

The result of limit monitoring is made available at the interconnectable output parameters:

- `OutAH_Act = 1`: Limit for high alarm reached or exceeded
- `OutAL_Act = 1`: Limit for low alarm reached or undershot
- `OutWH_Act = 1`: Limit for high warning reached or exceeded
- `OutWL_Act = 1`: Limit for low warning reached or undershot

- `OutTH_Act = 1`: Limit for high tolerance reached or exceeded
- `OutTL_Act = 1`: Limit for low tolerance reached or undershot

made available (see figure). The `SumMsgAct = 1` output parameter is also set when at least one limit value is reached or violated.

You can use `Feature.Bit29` Signaling limit violation (Page 181) to determine whether the output parameter triggers limit monitoring with the value "0" or "1".

You can use `Feature.Bit28` Disabling operating points (Page 156) to disable limit monitoring when message suppression is enabled (`MsgLock = 1`).

### Activating limit monitoring

Monitoring is always enabled using the input parameters:

- `OutAH_En = 1`: Monitoring of the high alarm limits
- `OutAL_En = 1`: Monitoring of the low alarm limits
- `OutWH_En = 1`: Monitoring of the high warning limits
- `OutWL_En = 1`: Monitoring of the low warning limits
- `OutTH_En = 1`: Monitoring of the high tolerance limits
- `OutTL_En = 1`: Monitoring of the low tolerance limits

**Predefinition:** When the block is installed, monitoring of the tolerance limits is disabled, meaning that the parameters are configured with 0. To activate monitoring, assign 1 to these parameters.

All other monitoring functions are enabled.

### Message suppression

The corresponding message is suppressed using the parameters:

- `OutAH_MsgEn = 0`: Alarm (high) messages are suppressed
- `OutAL_MsgEn = 0`: Alarm (low) messages are suppressed
- `OutWH_MsgEn = 0`: Warning (high) messages are suppressed
- `OutWL_MsgEn = 0`: Warning (low) messages are suppressed
- `OutTH_MsgEn = 0`: Tolerance (high) messages are suppressed
- `OutTL_MsgEn = 0`: Tolerance (low) messages are suppressed

The output of messages is not suppressed when the block is installed (all `xx_MsgEn` parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

### Operating in the faceplate

You can also change the limits using the faceplate. Refer to the Limit operation and display in the faceplate (Page 355) chapter for more on this.

### See also

User-configured message classes (Page 45)

### Limit monitoring of an additional analog value

### Limit monitoring of an additional analog value

Limit monitoring is performed for an additional analog value on the basis of the AV block, see Description of AV (Page 705) chapter.

You can monitor an additional analog value to the following high and low limits for alarms warnings and tolerances at the technologic block:

- `AV_AH_Lim`: Limit for high alarm
- `AV_AL_Lim`: Limit for low alarm
- `AV_WH_Lim`: Limit for high warning
- `AV_WL_Lim`: Limit for low warning
- `AV_TH_Lim`: Limit for the high tolerance
- `AV_TL_Lim`: Limit for the low tolerance

**Note**

The message classes Alarm, Warning and Tolerance are not valid for user-configured message classes. Please take into consideration the validity of terms for user-configured message classes (Page 45).

**Note**

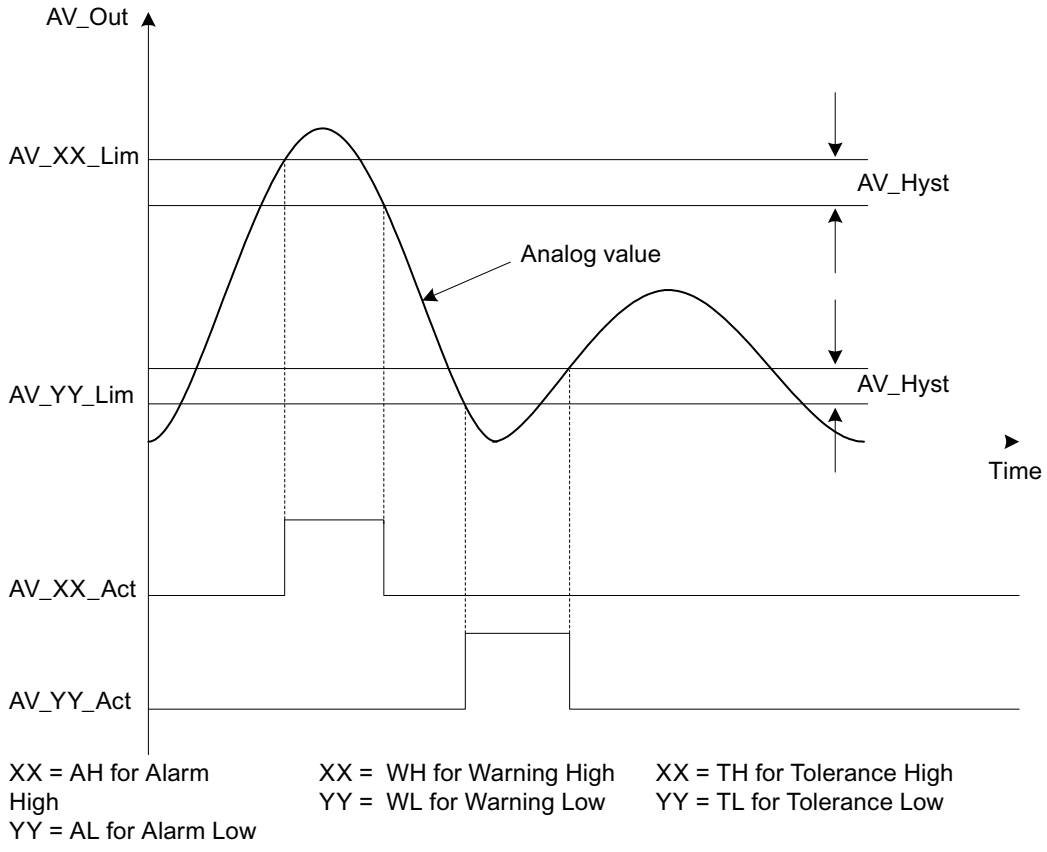
The AV block and the technologic block must be built into the same cyclic interrupt OB.

### Result of the limit monitoring

The result of limit monitoring is made available at the interconnectable output parameters of the AV block:

- `AV_AH_Act = 1:` Limit for high alarm reached or exceeded
- `AV_AL_Act = 1:` Limit for low alarm reached or undershot
- `AV_WH_Act = 1:` Limit for high warning reached or exceeded
- `AV_WL_Act = 1:` Limit for low warning reached or undershot
- `AV_TH_Act = 1:` Limit for high tolerance reached or exceeded
- `AV_TL_Act = 1:` Limit for low tolerance reached or undershot

made available (see figure).




You can use `Feature.Bit29` Signaling limit violation (Page 181) to determine whether the output parameter triggers limit monitoring with the value "0" or "1".

You can use `Feature.Bit28` Disabling operating points (Page 156) to disable limit monitoring when message suppression is enabled (`MsgLock = 1`).

### Activating limit monitoring

Monitoring is always enabled using the input parameters of the AV block:

- `AV_AH_En = 1`: Monitoring of the high alarm limits
- `AV_AL_En = 1`: Monitoring of the low alarm limits
- `AV_WH_En = 1`: Monitoring of the high warning limits
- `AV_WL_En = 1`: Monitoring of the low warning limits
- `AV_TH_En = 1`: Monitoring of the high tolerance limits
- `AV_TL_En = 1`: Monitoring of the low tolerance limits

**Predefinition:** When the block is installed, monitoring of the tolerance limits is disabled, meaning that the parameters are configured with 0. To activate monitoring, assign 1 to these parameters.

All other monitoring functions are enabled.

### Message suppression

The corresponding message is suppressed at the block AV using the parameters:

- `AV_AH_MsgEn = 0`: Alarm (high) messages are suppressed
- `AV_AL_MsgEn = 0`: Alarm (low) messages are suppressed
- `AV_WH_MsgEn = 0`: Warning (high) messages are suppressed
- `AV_WL_MsgEn = 0`: Warning (low) messages are suppressed
- `AV_TH_MsgEn = 0`: Tolerance (high) messages are suppressed
- `AV_TL_MsgEn = 0`: Tolerance (low) messages are suppressed

The output of messages is not suppressed when the block is installed (all `xx_MsgEn` parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

### Hysteresis

You can specify a hysteresis (`AV_Hyst`) at the technologic block for the limits, for example, to suppress signal flutter. Refer to the Limit monitoring with hysteresis (Page 104) chapter for more on this.

### Alarm delays

You can set alarm delays for incoming and outgoing alarms, warnings and tolerances. Refer to the Area of application of the alarm delays (Page 225) chapter for more on this.

### Operating in the faceplate

You can also change the limits and the hysteresis using the faceplate. Refer to the Limit operation and display in the faceplate (Page 355) chapter for more on this.

### See also

User-configured message classes (Page 45)

## Limit monitoring of the feedback

### Limit monitoring of position feedback

The position feedback of the manipulated variable can be monitored for the following high and low warning limits:

- `RbkWH_Lim`: Limit for high warning
- `RbkWL_Lim`: Limit for low warning

**Note**

The message classes Alarm, Warning and Tolerance are not valid for user-configured message classes. Please take into consideration the validity of terms for user-configured message classes (Page 45).

### Result of limit monitoring of the position feedback

The result of limit monitoring of the position feedback is made available at the interconnectable output parameters:

- `RbkWH_Act = 1`: High limit reached or exceeded
- `RbkWL_Act = 1`: Low limit reached or undershot

made available. The `SumMsgAct = 1` output parameter is also set when at least one limit value is reached or violated.

When the limits are reached or exceeded, messages that can be suppressed are output.

You can use `Feature.Bit29` Signaling limit violation (Page 181) to determine whether the output parameter triggers limit monitoring with the value "0" or "1".

You can use `Feature.Bit28` Disabling operating points (Page 156) to disable limit monitoring when message suppression is enabled (`MsgLock = 1`).

### Activating limit monitoring

Monitoring is always enabled using the input parameters:

- `RbkWH_En = 0`: Monitoring of the high warning limit is disabled
- `RbkWL_En = 0`: Monitoring of the low warning limit is disabled

**Predefinition:** When the block is installed, monitoring is enabled (default is 1).

### Message suppression

The corresponding message is suppressed using the parameters:

- `RbkWH_MsgEn = 0`: Messages from the high limit monitoring are suppressed
- `RbkWL_MsgEn = 0`: Messages from the low limit monitoring are suppressed

The output of messages is not suppressed when the block is installed (for example, `RbkWH_MsgEn = 1`). Messages can only be output if limit monitoring of the position feedback has been enabled.

### Hysteresis

You can specify a hysteresis (`RbkHyst`) for the limits, for example, to suppress signal flutter. Refer also to chapter Limit monitoring with hysteresis (Page 104) for more on this.

### Alarm delays (only for the PIDConR and MotSpdCL blocks)

You can set alarm delays for incoming and outgoing warnings. Refer to the Area of application of the alarm delays (Page 225) chapter for more on this.

### Operating in the faceplate

You can also change the limits and the hysteresis using the faceplate. Refer to the Limit operation and display in the faceplate (Page 355) chapter for more on this.

### See also

User-configured message classes (Page 45)

### Limit monitoring of setpoint, manipulated variable and control deviation

### Limit monitoring of setpoint, manipulated variable and error signal

The setpoint, manipulated variable and error signal can be monitored for the following high and low alarm limits:

- `ER_AH_Lim`: Limit for high alarm
- `ER_AL_Lim`: Limit for low alarm

**Note**

The message classes Alarm, Warning and Tolerance are not valid for user-configured message classes. Please take into consideration the validity of terms for user-configured message classes (Page 45).

### Result of the limit monitoring

The result of limit monitoring is made available at the interconnectable output parameters:

- `ER_AH_Act =1`: High limit violated (reached or exceeded)
- `ER_AL_Act =1`: Low limit (reached or undershot)

made available. The `SumMsgAct = 1` output parameter is also set when at least one limit value is reached or violated.

When the limits are reached or exceeded, messages that can be suppressed are output.

You can use `Feature.Bit29` Signaling limit violation (Page 181) to determine whether the output parameter triggers limit monitoring with the value "0" or "1".

You can use `Feature.Bit28` Disabling operating points (Page 156) to disable limit monitoring when message suppression is enabled (`MsgLock = 1`).

### Activating limit monitoring

Alarm monitoring is enabled using the input parameters:

- `ER_AH_En = 1`: Monitoring of the high alarm limit
- `ER_AL_En = 1`: Monitoring of the low alarm limit

**Predefinition:** When the block is installed, monitoring is disabled.

### Message suppression

The corresponding message is suppressed using the parameters:

- `ER_AH_MsgEn = 0`: Messages from the high limit monitoring are suppressed
- `ER_AL_MsgEn = 0`: Messages from the low limit monitoring are suppressed

The output of messages is not suppressed when the block is installed (for example, `ER_AH_MsgEn = 1`). Messages can only be output if limit monitoring has been enabled.

**Note**

With the MotSpdCL block, messages are only output if you have enabled `Feature.Bit5` (Alarm setpoint difference (Page 182)).

### Hysteresis

You can specify a hysteresis (`ER_Hyst`) for these limits, for example, in order to suppress signal flutter . Refer also to chapter Limit monitoring with hysteresis (Page 104) for more on this.

### Alarm delays

You can set alarm delays for incoming and outgoing alarms. Refer to the Area of application of the alarm delays (Page 225) chapter for more on this.

### Operating in the faceplate

You can also influence the limits and the hysteresis by means of the faceplate. Refer to the Limit operation and display in the faceplate (Page 355) chapter for more on this.

### See also

User-configured message classes (Page 45)

### Limit monitoring with hysteresis

### Limit monitoring with hysteresis

You can additionally define a hysteresis for all limit monitoring functions (parameter `xxx_Hyst`, xxx can, for example, be `PV` for the process value). You use the hysteresis, for example, to suppress signal flutter.

Enter the hysteresis as a physical variable at the block and faceplate (if you have the appropriate operator control permissions (WinCC)).

For the WinCC operator control permissions, refer to the help on WinCC.

## 2.1.3.4 Feedbacks

### Monitoring the feedbacks

### Feedback monitoring

You can use the following monitoring functions:

- Monitoring the start-up and stop characteristics for motors or the runtime of valves
- Monitoring the operation of motors or the maintenance of the position of valves
- Disabling feedback

This monitoring function is enabled via the `Monitor = 1` input.

Static and dynamic errors are reset by disabling the monitoring (`Monitor = 0`). If you reactivate monitoring during the plant runtime, only dynamic monitoring (`MonTiDynamic`) will be performed.

**Note**

In Manual mode with Reset functionality deactivated (`Feature.Bit31` = 0), you can control all the valves (including the motor valve but not the motors) despite the setting `MonSafePos` = 1 and with "Monitoring error" without reaching the neutral position.

### Monitoring the start-up and stop characteristics for motors or the runtime of valves

Monitoring of the startup characteristics is implemented using the parameter `MonTiDynamic`. The monitoring time specifies the period within which the feedback value, for example, `FbkStart` with motors, must be available in response to a control signal. If this is not the case, the text "Control error" is displayed in the standard view of the faceplate. An error message is generated at the same time. The block then goes to its neutral position. In the case of motors, this is always the stop state. With other blocks, this is an neutral position you have specified (`SafePos` parameter). The block signals this at the corresponding output parameter of the error message with 1, for example, with `MonDynErr = 1` for motors.

With the motors MotL, MotRevL, MotSpdL, MotSpdCL and the motor valve VlvMotL, the `Feature.Bit13` can be set to use a separate monitoring time `MonTiDyStop` for the stop behavior. The monitoring time for the stop behavior is displayed at the `MonDynStopErr` output.

Parameters are set in seconds.

### Monitoring the operation of motors or the maintenance of the position of valves

**The following applies for "Large" blocks:** Monitoring of the operation or the maintenance of the position of valves is implemented using the parameter `MonTiStatic`. The monitoring time specifies the period in which the feedback value can change its value briefly without an error message being output. An example would be a running motor with the feedback via the input parameter `FbkStart`. This parameter should be static in accordance with the control function. However, its value can change within the monitoring time. If the change in the `FbkStart` parameter takes longer than the monitoring time, the text "End position error" is displayed in the standard view of the faceplate. An error message is generated at the same time. The block then goes to its neutral position. In the case of motors, this is always the stop state. With other blocks, this is an neutral position you have specified (`SafePos` parameter). The block signals this at the corresponding output parameter of the error message with 1, for example, with `MonStaErr = 1` for motors.

Parameters are set in seconds.

**Note**

Please note that `MonTiDynamic ≥ MonTiStatic` and `MonTiDynamic ≥ SampleTime` have to be configured. If something is set outside these limits, the block always returns the respective limit at the input.

If `SampleTime` changes, `MonTiDynamic` may be tracked to the new value for `SampleTime`. `MonTiStatic` is tracked if `MonTiDynamic < MonTiStatic` changes. With `MonTiStatic = 0`, each feedback change without change of the control immediately results in a runtime error.

After a static monitoring error (e.g.: `MonStaErr` =1), an impulse output control is set to drive the motor or the valve in rest position even if the block is already in rest position.

When Monitoring Error occurs, `STOP` output pin value remains 0 until the error state got rectified and reset.

When `Trip`, `FaultExt`, `CSF`(with `Feature.Bit18` = 1) becomes active and error occurs, `STOP` outpin pin is set to 1, even at the error state and before reset.

**The following applies for "Small" blocks:** These blocks operate like "Large" blocks; however, the monitoring time is set to 0 within the block and cannot be changed. Any change is displayed immediately at the output parameter `MonStaErr` with 1.

### Disabling feedback for valves

You can also disable feedback completely. Please refer to section Disabling feedback for valves (Page 106) for further information.

**Note**

This function is only supported by "Large" blocks.

### Resetting the block in case of interlocks or errors

In the event of an interlock or error, the block has to be reset. Refer to the Resetting the block in case of interlocks or errors (Page 47) section for more on this.

### Disabling feedback for valves

### Disabling monitoring of feedback for valves

This function is only supported by "Large" blocks.

If you operate a block without feedback, use the parameter `NoFbkOpen = 1` bzw. `NoFbkClose = 1`. This means, for example, that you do not have any feedbacks for the opened state of the valve. Monitoring is thus disabled for this feedback. The feedback at the block is adjusted according to the control signal.

### Disabling monitoring of feedback for valves and motors

This function is only supported by "Large" blocks.

If you operate a block without feedback, use the parameter `NoFbkXxx` = 1. This means, for example, that you do not have any feedbacks for the opened state of the valve or running/ stopped state of the motor.

Monitoring is thus disabled for this feedback. The feedback at the block is adjusted according to the control signal and the other available feedbacks.

## 2.1.3.5 Motor protection function

### Motor protection function

The motor protection function is used to turn off the motor if there is thermal overload (`Trip = 0`, interconnectable input parameter).

If the motor is turned off by the motor protection function, a message (process control message) is generated. This is indicated in the faceplate by the "Motor protection" text. You can influence the reset using a various `Feature` bits. Refer to the Resetting the block in case of interlocks or errors (Page 47) section for more on this.

You can find more information in Section Influence of the signal status on the interlock (Page 112).

## 2.1.4 Interlocking functions

### 2.1.4.1 Interlocks

#### Interlocks at blocks

A maximum of three types of interlock can be used depending on the block. Three separate inputs named `Permit`, `Intlock` and `Protect` are available for these functions.

- **Activation enable ("Permission"):** The activation enable makes it possible to switch the motor in the direction which is enabled. The activation disable is displayed with a padlock in the standard view. The motor cannot switch in the direction which is disabled.
- **Interlock without reset ("Interlock"):** An active interlock condition puts the block to the neutral position (`Intlock` = 0). After the interlock condition has gone, the currently active control function becomes active again in automatic or local mode. In manual mode, the faceplate can be operated again after the interlock condition has gone.
- **Interlock with reset ("Protection"):** An active interlock condition puts the block to the neutral position (input `Protect` = 0). Once the interlock condition is cleared, a reset is necessary. Refer to the Resetting the block in case of interlocks or errors (Page 47)section for more information.
- **Permission for safe position ("Permit neutral pos." or "Permission stop"):**
  - The activation enable ("PermitSafePos = 0") prevents stop command to the motor.
  - The activation enable ("PermitSafePos = 0") prevents the control in safe position direction for VlvL & VlvAnL.
  - The activation enable ("PermitSafePos = 0") & Feature2.Bit16 = 0 prevents the control in safe position direction for VlvMotL & VlvPosL.
  - The activation enable ("PermitSafePos = 0") & Feature2.Bit16 = 1 prevents the stop command for VlvMotL & VlvPosL.

**Note**

**Interlocks for "small" blocks and controller blocks**

Ensure that "small" blocks and controller blocks PIDConL and PIDConR have only the `Intlock` parameter. The other two interlocks are not included in these block variants.

**Permission stop for VlvPosL**

Permission stop does not restrict manipulated variable when position control is active.

**Permission stop for MotRevL & MotSpdCL**

Permission stop does not restrict the direct changeover between forward and revrese direction (`Feature.Bit7` = 1)

**Interlock dependent on direction, position or speed**

The blocks MotRevL, MotSpdL, VlvMotL, VlvPosL, VlvAnL and MotSpdCL with `Feature2.Bit16` = 1 (see section Separate interlock for each direction, position, or speed (Page 180)) have inputs for interlock functionality with direction:

| **MotRevL, MotSpdCL (`Feature2.Bit16` = 1)** | **MotSpdL (`Feature2.Bit16` = 1)** | **VlvMotL, VlvPosL, VlvAnL (`Feature2.Bit16` = 1)** |
|---|---|---|
| `Permit`: Enable forward | `Permit`: Enable speed1 | `Permit`: Enable open |
| `PermRev`: Enable reverse | `PermSp2`: Enable speed2 | `PermCls`: Enable close |
| `Intlock`: Interlock forward | `Intlock`: Interlock speed1 | `Intlock`: Interlock open |
| `IntlRev`: Interlock reverse | `IntlSp2`: Interlock speed2 | `IntlCls`: Interlock close |
| `Protect`: Protection forward | `Protect`: Protection speed1 | `Protect`: Protection open |
| `ProtRev`: Protection reverse | `ProtSp2`: Protection speed2 | `ProtCls`: Protection close |

**Interlock function dependent on direction (`Feature2.Bit16` = 1) for MotRevL:**

- **Activation enable forward/reverse:** The activation enable makes it possible to switch the motor in the direction which is enabled. The activation disable is displayed with a padlock in standard view. The motor cannot switch in the direction which is disabled.
- **Interlock forward/reverse without reset:** An active interlock condition only puts the block in the stop position, when the motor is running in direction which is interlocked. Once interlock condition is cleared, the currently active control function becomes active again in automatic or local mode. In manual mode, the faceplate can be operated again in this direction.
- **Interlock forward/reverse with reset:** An active interlock condition only puts the block in the stop position, when the motor is running in the direction which is interlocked. Once the interlock condition is cleared, a reset is necessary. Interlock forward and reverse can be reset separately. Refer to the Resetting the block in case of interlocks or errors (Page 47) section for more information.

**Interlock function dependent on speed (`Feature2.Bit16` = 1) for MotSpdL:**

- **Activation enable speed1/speed2:** The activation enable makes it possible to switch the motor in the speed which is enabled. The activation disable is displayed with a padlock in standard view. The motor cannot switch in the speed which is disabled.
- **Interlock speed1/speed2 without reset**: An active interlock condition only puts the block in the stop position, when the motor is running with the peed which is interlocked. Once interlock condition is cleared, the currently active control function becomes active again in automatic or local mode. In manual mode, the faceplate can be operated again in this speed.
- **Interlock speed1/speed2 with reset:** An active interlock condition only puts the block in the stop position, when the motor is running with the speed which is interlocked. Once the interlock condition is cleared, a reset is necessary. Interlock speed1 and speed2 can be reset separately. Refer to the Resetting the block in case of interlocks or errors (Page 47) section for more information.

**Interlock function dependent on position (`Feature2.Bit16` = 1) for VlvMotL, VlvPosL, VlvAnL:**

- **Activation enable open/close:** The activation enable makes it possible to move the valve in the direction of the position which is enabled. The activation disable is displayed with a padlock in standard view. The valve cannot move in the direction of the position which is disabled.
- **Interlock open/close without reset:** An active interlock condition only puts the block in the neutral position, when the valve is moving to a position which is interlocked. If the interlocked position is the neutral position open or close, valve goes only to stop position. Once interlock condition is cleared, the currently active control function becomes active again in automatic or local mode. In manual mode, the faceplate can be operated again from this position.
- **Interlock open/close with reset:** An active interlock condition only puts the block in the neutral position, when the valve is moving to a position which is interlocked. If the interlocked position is the neutral position open or close, valve goes only to stop position. Once the interlock condition is cleared, a reset is necessary. Interlock open and close can be reset separately. Refer to the Resetting the block in case of interlocks or errors (Page 47) section for more information.

## Display of the interlock in the faceplate and in the block icon

The interlock state is visualized in the faceplate and in the block icon by a status display (padlock) as follows:

- Open padlock: No interlock pending
- Closed padlock: One or more interlocks are pending
- No padlock: Individual interlocks are not active
  - `Perm_En` = 0 or `Permit.ST` = 16#FF:
    The input parameter `Permit` has no effect, the button in the faceplate is invisible.
  - `PermRevEn`/`PermClsEn`/`PermSp2En` = 0 or `PermRev.ST`/`PermCls.ST`/`PermSp2.ST` = 16#FF or `Feature2.Bit16` = 0:
    The input parameter `PermRev`/`PermCls`/`PermSp2` has no effect, the button in the faceplate is invisible.
  - `Prot_En` = 0 or `Protect.ST` = 16#FF:
    The input parameter `Protect` has no effect, the button in the faceplate is invisible.
  - `ProtRevEn`/`ProtClsEn`/`ProtSp2En` = 0 or `ProtRev.ST`/`ProtCls.ST`/`ProtSp2.ST` = 16#FF or `Feature2.Bit16` = 0:
    The input parameter `ProtRev`/`ProtCls` has no effect, the button in the faceplate is invisible.
  - `Intl_En` = 0 or `Intlock.ST` = 16#FF:
    The input parameter `Intlock` has no effect, the button in the faceplate is invisible
  - `IntlRevEn`/`IntlClsEn`/`IntlSp2En` = 0 or `IntlRev.ST`/`IntlCls.ST`/`IntlSp2.ST` = 16#FF or `Feature2.Bit16` = 0:
    The input parameter has no effect, the button in the faceplate is invisible.
  - PermSf_En = 0 or PermitSafePos.ST = 16#FF or for VlvAnL when SafePos = 2 :
    The input parameter PermitSafePos has no effect, the button in the faceplate is invisible.

The block icon indicates the prioritized group status according to the active operating state. See also the section Forming the group status for interlock information (Page 113) for more on this.

The faceplate visualizes the state of each interlock type separately.

The padlock is not shown in the block icon if all parameters for enabling the button are set to 0 or all parameters have the signal status `16#FF`.

**Note**

Motors and values are not put into the neutral position if one of the interlock inputs is active (for example `Intlock = 0`) and the corresponding signal status is `16#FF` (`Intlock.ST = 16#FF`).

### Influence of the signal status on the interlock

See also the section Influence of the signal status on the interlock (Page 112) for more on this.

### Outputting "Interlock active" using the `LockAct` parameter

If an interlock is set at the parameter:

- `Intlock`
- `Permit`
- `Protect`
- `IntlRev`/`IntlCls`/`IntlSp2` (only for blocks with directional interlock and `Feature2.Bit16` = 1)
- `PermRev`/`PermCls`/`PermSp2` (only for blocks with directional interlock and `Feature2.Bit16` = 1)
- `ProtRev`/`ProtCls`/`ProtSp2` (only for blocks with directional interlock and `Feature2.Bit16` = 1)
- `Trip` (only for motors and motor valves),
- PermitSafePos

the `LockAct` parameter is set automatically to active (=1). The parameter `LockAct` is set to 0 if the interlock is no longer present and those interlocks which require acknowledgement have been acknowledged.

You can bypass the interlock using `BypProt = 1` in local mode and during simulation. This also makes `LockAct = 0`.

**Note**

The `LockAct` parameter is not set despite a pending interlock, if a value in the block is forced. See also the section Forcing operating modes (Page 45) for more on this.

### Messaging

No messages are assigned to the interlock types. However, if you want to have a message when an interlock condition is violated, you can use the freely interconnectable input parameters to generate the messages. See also the section Generating instance-specific messages (Page 230) for more on this.

### See also

Neutral position for motors, valves and controllers (Page 54)

## 2.1.4.2 Disabling interlocks

### Disabling individual interlocks

You can disable the block interlocks that are implemented using the input parameters `Intlock`, `Protect`, `Permit`, `IntlRev/Cls/Sp2`, `ProtRev/Cls/Sp2`,and `PermRev/Cls/Sp2`.

If you want to disable the block interlock, you have to set the following parameters:

- `Perm_En = 0` or `Permit.ST = 16#FF`:
  The input parameter `Permit` has no effect.
- `PermRevEn/PermClsEn/PermSp2En = 0` or `PermRev.ST/PermCls.ST/PermSp2.ST = 16#FF` or `Feature2.Bit16` = 0:
  The input parameter `PermRev/PermCls/PermSp2` has no effect.
- `Prot_En = 0` or `Protect.ST = 16#FF`:
  The input parameter `Protect` has no effect.
- `ProtRevEn/ProtClsEn/ProtSp2En = 0` or `ProtRev.ST/ProtCls.ST/ProtSp2.ST = 16#FF` or `Feature2.Bit16` = 0:
  The input parameter `ProtRev/ProtCls/ProtSp2` has no effect.
- `Intl_En = 0` or `Intlock.ST = 16#FF`:
  The input parameter `Intlock` has no effect.
- `IntlRevEn/IntlClsEn/IntlSp2En = 0` or `IntlRev.ST/IntlCls.ST/IntlSp2.ST = 16#FF` or `Feature2.Bit16` = 0:
  The input parameter `IntlRev/IntlCls/IntlSp2` has no effect.
- PermSf_En = 0 or PermitSafePos.ST = 16#FF or for VlvAnL when SafePos = 2 :
  The input parameter PermitSafePos has no effect.

**Note**

**"Small" blocks have only the `Intlock` parameter**

With "Small" blocks, you can only assign parameters to "Interlock without reset" (input parameter `Intlock` or for the deactivation of the interlock for the input parameter `Intl_En`).

### Disabling of all the interlocks (only for local operation and for simulation)

You can use the input parameter `BypProt = 1` to disable all the interlocks, irrespective of the parameter assignment of the individual interlock, in local mode as well as for the "simulation" function.

## 2.1.4.3 Influence of the signal status on the interlock

### Influence of the signal status on the interlock

There are three ways in which the signal status affects the interlocks:

- Simulation signal status
- Signal status "Bad, device related" (value 16#00) or "Bad, process related" (value 16#28)
- Signal status ≠ "Simulation" and "Bad, device related"

### "Simulation" signal status

An interlock signal is displayed differently depending on the "Separate evaluation for excluded and simulated interlock signals (Page 162)" function and the status of the interlock (see "Forming the group status for interlock information (Page 113)").

### Signal status "Bad, device related" (value 16#00) or "Bad, process related" (value 16#28)

An interlock signal with this status is always processed as an active interlock signal in the block and displayed with the following icons in the faceplate:



for 16#00 or



for 16#28 and



A motor protection signal (`Trip` parameter) with signal status 16#00 or 16#28 is used to activate motor protection. This is indicated by "Motor protection" in the standard view of the faceplates.

A torque-monitoring signal (`TorqOpen`, `TorqClose` parameters) with signal status 16#00 or 16#28 is used to activate torque monitoring for motor valve VlvMotL.

### Signal status ≠ "Simulation" as well as "Bad, device related" and "Bad, process related"

Only signal states "Simulation", "Bad, device related" and "Bad, process related" have an effect on the processing in the block; all others are only displayed with their relevant icon in the faceplate.

### 2.1.4.4 Forming the group status for interlock information

#### Forming the group status for interlock information

A group status for interlock information is required for:

- Interlock state for:
  - Interlocked
  - Not interlocked
  - Disabled

#### Group status for the interlock state

All the effective interlock states are combined and displayed in the block icon. The interlock states are displayed with the following prioritization:

1. Function interlocked, shown in the block icon with a closed padlock



2. Function not interlocked, shown in the block icon as an open padlock



3. Function disabled, shown in the block icon as a crossed-out, closed padlock



#### Overview: Display for the interlock status in faceplate

If the Separate evaluation for excluded and simulated interlock signals (Page 162) function is not enabled (`Feature` bit =0), the bypass signals in the structures Permission, Protection, and Interlock are not evaluated.

Interlocks are shown in the faceplate as follows:

| **Parameter: BypProt** | **Parameter: Perm_En[2] or Prot_En[2] or Intl_En [2] or PermSf_En** | **Value: Permission[2] or Protection[2] or Intlock [2] PermitSafe-Pos** | **Status: Permission[2] or Protection[2] or Intlock[2] PermitSafePos** | **Button** | **Icon** | **Icon** |
|---|---|---|---|---|---|---|
| x | x | X | 16#FF | - | - | - |
| x | 0 | x | x | - | - | - |
| 0 | 1 | 1 | 16#00 | Visible |  |  |
| 0 | 1 | 1 | 16#28 | Visible |  |  |

| **Parameter: BypProt** | **Parameter: Perm_En[2] or Prot_En[2] or Intl_En [2] or PermSf_En** | **Value: Permission[2] or Protection[2] or Intlock [2] PermitSafe-Pos** | **Status: Permission[2] or Protection[2] or Intlock[2] PermitSafePos** | **Button** | **Icon** | **Icon** |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 16#60 | Visible |  |  |
| 0 | 1 | 1 | 16#60 | Visible |  |  |
| 0 | 1 | 0 | ≠ 16#FF | Visible |  | Status based on priority |
| 0 | 1 | 1 | ≠ (16#FF, 16#60, 16#00, 16#28) | Visible |  | Status based on priority |
| 1 (for local mode and simulation only) | 1 | x | ≠ 16#FF | Visible |  | Status based on priority |

Comments on table:

- X: The value is irrelevant for the display of the icon.
- [2]: Blocks with directional interlock and `Feature2.Bit16` = 1: "`Perm_En`", "`Intl_En`", "`Prot_En`", "Enable", "Interlock" and "Protection" also include the direction

**Note**

If values are forced in the block, this is indicated in the block icon by a crossed-out, closed padlock.

If the Separate evaluation for excluded and simulated interlock signals (Page 162) function is enabled (`Feature` bit =1), the bypass signals in the structures Permission, Protection, and Interlock are evaluated. The bypass signals can be read from the upstream interlock block at the output Bypass.

Excluded interlock signals are shown in the faceplate as follows:

| **Parameter: Perm_En[2] or Prot_En[2] or Intl_En[2]** | **Status: Permission[2] or Protection[2] or Intlock[2]** | **Bypass: Permission[2] or Protection[2] or Intlock[2]** | **Button** | **Icon** | |
|---|---|---|---|---|---|
| x | 16#FF | x | - | - | |
| 0 | x | x | - | - | |
| 1 | ≠ 16#FF | 0 | Visible | | 1) |
| 1 | ≠ 16#FF | 1 | Visible |  | 1) |

Comments on table:

- X: The value is irrelevant for the display of the icon.
- 1): Icon corresponding to the status of the interlock signal (for 16#80 no icon)
- 2): Blocks with directional interlock and `Feature2.Bit16` = 1: "`Perm_En`", "`Intl_En`", "`Prot_En`", "Enable", "Interlock" and "Protection" also include the direction

### 2.1.4.5 Rapid stop for motors

#### Rapid stop for motors

Rapid stop has the highest priority in all operating modes (manual and automatic mode as well as local mode) and operating states (such as the forcing of states). It is activated via the faceplate. This depends on the setting at the `Feature` bit Enabling rapid stop via faceplate (Page 179).

**Note**

**"Small" blocks do not feature rapid stop.**

The "rapid stop" function is supported only by "Large" blocks.

You issue the command for rapid stop state using the `RapidStp = 1` input parameter.

When you click on the "Rapid Stop" button in the faceplate, the drive stops immediately, shown as follows in the faceplate:




The `R_StpAct = 1` output parameter is set to implement the rapid stop function for local mode. You need to interconnect this parameter with the corresponding channel block and in the I/O to realize the rapid stop function in the hardware.

Rapid stop is unlocked for all operating modes using the "Reset" button in the faceplate (`RstOp = 1`); in CFC it is unlocked using the `RstLi = 1` input parameter. In automatic mode, the unlocking can also be performed via a 0-1 edge transition in the control if `Feature.Bit9` = 1.

Rapid stop can be selected even with the motor in stop state. In this case, the motor start is prevented.

### 2.1.4.6 Bypassing signals

#### Bypassing signals

With this function, you can flag a block as bypassed. If the bypass function is active, the output parameter `BypassAct` will be set from the block. With a connection from `BypassAct` to an input parameter `BypLixx` of an interlock block, you can bypass the corresponding input of the interlock block.

If the block is set to bypass, the bypass information bit of the process output and the limit switch points listed in the following table are set. If you connect one of this listed parameter with bypass information bit to an interlock input `Inx`, you can use the bypass information bit to switch the interlock signal in bypass. For this in the interlock block the corresponding bypass enable bit `BypEn.Inx` and the `Feature.Bit2` Separate evaluation for excluded and simulated interlock signals (Page 162) have to set.

Additionally, you can define over the `Feature` bit Substitution value is active if the block is in bypass (Page 199) if the block shall use the process value or a substitution value in case of an active bypass.

| Block | Process value/ signal | Process value/ signal | Parameter with bypass information bits |
|---|---|---|---|
| MonAnL | `PV_Out` | `BypPV` | `PV_Out`, `PV_AH_Act`, `PV_WH_Act`, `PV_TH_Act`, `PV_AL_Act`, `PV_WL_Act`, `PV_TL_Act`, `GradHUpAct`, `GradHDnAct`, `GradLAct` |
| PIDConL, PID-ConR, and PID-StepL | `PV_Out` | `BypPV` | `PV_Out`, `PV_AH_Act`, `PV_WH_Act`, `PV_TH_Act`, `PV_AL_Act`, `PV_WL_Act`, `PV_TL_Act`, `ER_AH_Act`, `ER_AL_Act` |
| MonDiL | `Out` | `BypIn` | `Out` |
| TotalL | `Out` | `BypOut` | `Out`, `OutAH_Act`, `OutWH_Act`, `OutTH_Act`, `OutAL_Act`, `OutWL_Act`, `OutTL_Act` |

With the parameter `BypLiOp`, you can decide whether the switching on/off will be done by the input parameters which are connected in CFC or by the faceplate from an operator.

- `BypLiOp` = 0: In the "Parameter view", the operator can switch on/off the bypass function over the parameter `BypPVOp` or `RstBypOp`. If `BypLock` = 1, you cannot make any change to the parameters `BypPVOp` or `ResBypOp`. The button for switching the bypass functionality on/off in the "Parameter view" is deactivated.
- `BypLiOp` = 1: With a connection of the parameter `BypPVLi` or `RstBypLi`, the bypass function will be switched on/off.

## 2.1.5 Form signal status

### 2.1.5.1 Forming and outputting signal status for blocks

#### General information on forming and outputting the signal status

The process values of the function blocks are generated and transferred along with a signal status as a structured variable. This contains a statement about the signal quality. The function blocks determine the appropriate signal status for their process outputs depending on the signal status of the process inputs, which are involved in calculating the process outputs. If multiple process inputs are involved in calculating a process output, the signal status is formed according to prioritization defined by function block groups. The highest priority is the signal status with the value 0.

The blocks are grouped into the following function block groups:

- Technologic blocks (Page 117)
- Digital logic blocks (Page 119)
- Analog logic blocks (Page 120)
- Redundancy blocks (Page 121)
- Blocks with configurable status prioritization (Page 122)
- Interlock blocks (Page 124)
- Mathematical blocks (Page 126)
- PCS 7 channel blocks (Page 127)
- Channel blocks for field devices (Page 127)

All blocks of a group use the same priority specifications and form the signal status of the process outputs based on them.

**Note**

The status / quality of control inputs for logic functions and parameters have no influence on the status / quality of process values and logic functions of the blocks.

The status / quality of process values inherit the results of mathematic and logic functions, which are directly related to the process value.

The status/quality of process values immediately inherit the results of monitoring and limiting functions directly related to the process value.

### 2.1.5.2 Forming and outputting the signal status for technologic blocks

#### Forming the signal status for technology blocks

For more general information on forming the status signal, refer to the section: Forming and outputting signal status for blocks (Page 116).

In technology blocks, a group status is formed from the input parameters (see description of the relevant blocks) according to the priority table below (highest priority is 0). This group status is displayed in the status bar of the faceplate and of the block icon.

The group status is set to 16#68 (Uncertain, device related) with an undefined signal status at a control input, which is involved in the formation of the group status.

| Signal status icon | Priority | Value | Meaning |
|---|---|---|---|
|  | 0 | 16#60 | Manipulated value (for example, substitute value, simulation, last valid value) |
|  | 1 | 16#00 | Bad, device related |
|  | 2 | 16#28 | Bad, process related |

| Signal status icon | Priority | Value | Meaning |
|---|---|---|---|
|  | 3 | 16#68 | Uncertain, device related |
|  | 4 | 16#78 | Uncertain, process related |
|  | 5 | 16#A4 | Maintenance request |
|  | 6 | 16#80 | Good |

Interconnectable output parameters for limits (for example, `PV_AH_Act`) that can be influenced directly by an interconnectable input parameter (for example, `PV`) inherit the status from the associated output parameter (for example, `PV_Out`).

If an output parameter for limits is influenced directly by several interconnectable input parameters (limit monitoring), it receives the status of the input parameter with the highest priority (see overview above). Thus, for example, the control deviation is formed from the setpoint (`SP`) and the process value (`PV`). The output parameter for limits `ER_AH_Act`, for example, which signals an active violation of the high limit for the error signal, has a signal status based on the group status formed from the process value and setpoint (`ER`).

### Evaluation of the signal status in case of interlocks in technology blocks

The signal states of the interconnectable input parameters of the interlocking and protective signals are treated exactly like process values with the following exceptions:

- The signal status is displayed in the faceplate at the buttons for calling the series-connected interlocking blocks.
- If the input signal has "Simulation" status and the input signal is therefore inactive (for example, `Protect` = 1), the input signal in this case is interpreted as a bypassed signal and is displayed by the icon for bypassing:

- If the input signal has "Simulation" status and the input signal is therefore inactive (for example, `Protect` = 0), the input signal in this case is interpreted as a simulated signal and is displayed by the icon for simulation:

- If the input signal has the "Bad, device related" or "Bad, process related" signal status, this is evaluated as an active input signal, regardless of its value, i.e. safety interlock signal (`Protect` = 1) that is inactive due to its value, triggers a safety interlock when its status is "Bad, device related" or "Bad, process related".

### Display of the signal status in the faceplate and block icon for technology blocks

The signal status is displayed for each individual input parameter in the faceplate next to the process values or the interlock buttons. The group status is displayed in the block icon and in the group display of the faceplate.

**Note**

The interlocks and additional values are not included in the formation of the group status.

## 2.1.5.3 Forming and outputting the signal status of digital logic blocks

### Forming the signal status of digital logic blocks

For more general information on forming the status signal, refer to the section: Forming and outputting signal status for blocks (Page 116).

The status for the result of the output is formed within digital logic blocks from all input parameters, according to the following priority table (highest priority is 0)

| Signal status icon | Priority | Value | Meaning |
|---|---|---|---|
|  | 0 | 16#00 | Bad, device related |
|  | 1 | 16#28 | Bad, process related |
|  | 2 | 16#60 | Manipulated value (for example, substitute value, simulation, last valid value) |
|  | 3 | 16#68 | Uncertain, device related |
|  | 4 | 16#78 | Uncertain, process related |
|  | 5 | 16#A4 | Maintenance request |
|  | 6 | 16#80 | Good |

The group status is set to 16#68 (uncertain, device-related), with an undefined signal status at a control input that is involved in forming the group status.

A signal status 16#FF at a control input is not used for the calculation of the group status. If all relevant control inputs are 16#FF, the group status is 16#80.

If only one process input is decisive for calculating the output value, the status of the process input is transferred to the status of the output.

**Special notes for the Andxx blocks**

- If the output value is 1, it has the signal status with the highest priority of all input signals.
- If the output value is 0, it has the signal status with the lowest priority of all input signals, which have a value of 0.

**Special notes for the Orxx blocks**

- If the output value is 1, it has the signal status with the lowest priority of all input signals, which have a value of 1.
- If the output value is 0, it has the signal status with the highest priority of all input signals.

**Special notes for the Xor04 block**

The worst signal status of all input parameters is always selected and output with the `Out` output parameter.

### 2.1.5.4 Forming and outputting the signal status of analog logic blocks

#### Forming the signal status of analog logic blocks

For more general information on forming the status signal, refer to the section: Forming and outputting signal status for blocks (Page 116).

The signal status of the Out output value within the block is taken directly from the `Inx` input value.

#### Special notes for the CompAn02 block

This block evaluates the signal status of the two input parameters `In1` and `In2` as shown in the following table.

| Signal status icon | Priority | Value | Meaning |
|---|---|---|---|
|  | 0 | 16#60 | Manipulated value (for example, substitute value, simulation, last valid value) |
|  | 1 | 16#00 | Bad, device related |
|  | 2 | 16#28 | Bad, process related |
|  | 3 | 16#68 | Uncertain, device related |
|  | 4 | 16#78 | Uncertain, process related |
|  | 5 | 16#A4 | Maintenance request |
|  | 6 | 16#80 | Good |

### 2.1.5.5 Forming and outputting the signal status of redundancy blocks

#### Forming the signal status of redundancy blocks RedAn02 and RedDi02

For more general information on forming the status signal, refer to the section: Forming and outputting signal status for blocks (Page 116).

The signal status is evaluated according to the following priority:

| Signal status icon | Priority | Value | Meaning |
|---|---|---|---|
|  | 0 | 16#60 | Manipulated value (for example, substitute value, simulation, last valid value) |
|  | 1 | 16#80 | Good |
|  | 2 | 16#A4 | Maintenance request |
|  | 3 | 16#78 | Uncertain, process related |
|  | 4 | 16#68 | Uncertain, device related |
|  | 5 | 16#28 | Bad, process related |
|  | 6 | 16#00 | Bad, device related |

The output value is generated based on the signal status of the process value with the above table of priorities. In addition, the output parameters `SimAct`, `Uncertain`, and `LossRed` are still set according to the signal status.

| In1.ST | In2.ST | Out.ST | Out.Value | SimAct.Value | Uncertain.Value | LossRed.Value |
|---|---|---|---|---|---|---|
| 16#80 | 16#80 | 16#80 | In1 | 0 | 0 | 0 |
| 16#80 | 16#60 | 16#60 | In2 | 1 | 1 | 0 |
| 16#80 | 16#A4 | 16#80 | In1 | 0 | 1 | 0 |
| 16#80 | 16#78 | 16#80 | In1 | 0 | 1 | 0 |
| 16#80 | 16#68 | 16#80 | In1 | 0 | 1 | 0 |
| 16#80 | 16#28 | 16#68 | In1 | 0 | 1 | 1 |
| 16#80 | 16#00 | 16#68 | In1 | 0 | 1 | 1 |
| 16#60 | 16#80 | 16#60 | In1 | 1 | 1 | 0 |
| 16#60 | 16#60 | 16#60 | In1 | 1 | 1 | 0 |
| 16#60 | 16#A4 | 16#60 | In1 | 1 | 1 | 0 |
| 16#60 | 16#78 | 16#60 | In1 | 1 | 1 | 0 |
| 16#60 | 16#68 | 16#60 | In1 | 1 | 1 | 0 |
| 16#60 | 16#28 | 16#60 | In1 | 1 | 1 | 1 |
| 16#60 | 16#00 | 16#60 | In1 | 1 | 1 | 1 |
| 16#A4 | 16#80 | 16#80 | In2 | 0 | 1 | 0 |

| In1.ST | In2.ST | Out.ST | Out.Value | SimAct.Value | Uncertain.Value | LossRed.Value |
|---|---|---|---|---|---|---|
| 16#A4 | 16#60 | 16#60 | In2 | 1 | 1 | 0 |
| 16#A4 | 16#A4 | 16#A4 | In1 | 0 | 1 | 0 |
| 16#A4 | 16#78 | 16#A4 | In1 | 0 | 1 | 0 |
| 16#A4 | 16#68 | 16#A4 | In1 | 0 | 1 | 0 |
| 16#A4 | 16#28 | 16#68 | In1 | 0 | 1 | 1 |
| 16#A4 | 16#00 | 16#68 | In1 | 0 | 1 | 1 |
| 16#78 | 16#80 | 16#80 | In2 | 0 | 1 | 0 |
| 16#78 | 16#60 | 16#60 | In2 | 1 | 1 | 0 |
| 16#78 | 16#A4 | 16#A4 | In2 | 0 | 1 | 0 |
| 16#78 | 16#78 | 16#78 | In1 | 0 | 1 | 0 |
| 16#78 | 16#68 | 16#78 | In1 | 0 | 1 | 0 |
| 16#78 | 16#28 | 16#68 | In1 | 0 | 1 | 1 |
| 16#78 | 16#00 | 16#68 | In1 | 0 | 1 | 1 |
| 16#68 | 16#80 | 16#80 | In2 | 0 | 1 | 0 |
| 16#68 | 16#60 | 16#60 | In2 | 1 | 1 | 0 |
| 16#68 | 16#A4 | 16#A4 | In2 | 0 | 1 | 0 |
| 16#68 | 16#78 | 16#78 | In2 | 0 | 1 | 0 |
| 16#68 | 16#68 | 16#68 | In1 | 0 | 1 | 0 |
| 16#68 | 16#28 | 16#68 | In1 | 0 | 1 | 1 |
| 16#68 | 16#00 | 16#68 | In1 | 0 | 1 | 1 |
| 16#28 | 16#80 | 16#68 | In2 | 0 | 1 | 1 |
| 16#28 | 16#60 | 16#60 | In2 | 1 | 1 | 1 |
| 16#28 | 16#A4 | 16#68 | In2 | 0 | 1 | 1 |
| 16#28 | 16#78 | 16#68 | In2 | 0 | 1 | 1 |
| 16#28 | 16#68 | 16#68 | In2 | 0 | 1 | 1 |
| 16#28 | 16#28 | 16#28 | In1 | 0 | 1 | 1 |
| 16#28 | 16#00 | 16#28 | In1 | 0 | 1 | 1 |
| 16#00 | 16#80 | 16#68 | In2 | 0 | 1 | 1 |
| 16#00 | 16#60 | 16#60 | In2 | 1 | 1 | 1 |
| 16#00 | 16#A4 | 16#68 | In2 | 0 | 1 | 1 |
| 16#00 | 16#78 | 16#68 | In2 | 0 | 1 | 1 |
| 16#00 | 16#68 | 16#68 | In2 | 0 | 1 | 1 |
| 16#00 | 16#28 | 16#28 | In2 | 0 | 1 | 1 |
| 16#00 | 16#00 | 16#00 | In1 | 0 | 1 | 1 |

### 2.1.5.6 Forming and outputting the signal status for blocks with configurable status prioritization

#### Forming the signal status for blocks with configurable status priority

For more general information on forming the status signal, refer to the section: Forming and outputting signal status for blocks (Page 116).

The SelPrio parameter is used for this block to define the priority setting for linking the individual states. You have the option between the following specifications:

| Priority | SelPrio = 0 | SelPrio = 1 | SelPrio = 2 | SelPrio = 3 | SelPrio = 4 | SelPrio = 5 | SelPrio = 6 | SelPrio = 7 |
|---|---|---|---|---|---|---|---|---|
| 0 |  16#60 |  16#80 |  16#A4 |  16#80 |  16#A4 |  16#60 |  16#80 |  16#60 |
| 1 |  16#00 |  16#00 |  16#00 |  16#A4 |  16#80 |  16#80 |  16#A4 |  16#80 |
| 2 |  16#28 |  16#28 |  16#28 |  16#60 |  16#60 |  16#A4 |  16#78 |  16#A4 |
| 3 |  16#68 |  16#68 |  16#68 |  16#00 |  16#00 |  16#00 |  16#68 |  16#78 |
| 4 |  16#78 |  16#78 |  16#78 |  16#28 |  16#28 |  16#28 |  16#28 |  16#68 |
| 5 |  16#A4 |  16#60 |  16#60 |  16#68 |  16#68 |  16#68 |  16#00 |  16#28 |
| 6 |  16#80 |  16#A4 |  16#80 |  16#78 |  16#78 |  16#78 |  16#60 |  16#00 |

## Special notes for the MuxAn03 block

The status priority for this block, which is used by the block to process the status of the PV1 ... PV3 process value inputs, can be set with the SelPrio parameter.

**Note**

The parameter SelPrio can take a value from 0 to 7 . SelPrio = 6 is set by default.

If you enter a value greater than 7, the setting for 7 is used. If you enter a value lower than 0, the setting for 0 is used.

For information on evaluating the status of the process value inputs, refer to the section MuxAn03 functions (Page 2331).

### 2.1.5.7 Forming and outputting the signal status for interlock blocks

## Forming the signal status for interlock blocks

For more general information on forming the status signal, refer to the section: Forming and outputting signal status for blocks (Page 116).

The block determines the signal status of the output signal, based on the signal status of the input values from the configured logical operation according to the following table (highest priority is 0):

| Signal status icon | Priority | Value | Meaning |
|---|---|---|---|
|  | 0 | 16#00 | Bad, device related |
|  | 1 | 16#28 | Bad, process related |
|  | 2 | 16#60 | Manipulated value (for example, substitute value, simulation, last valid value) |
|  | 3 | 16#68 | Uncertain, device related |
|  | 4 | 16#78 | Uncertain, process related |
|  | 5 | 16#A4 | Maintenance request |
| No icon | 6 | 16#80 | Good |
| No icon | - | 16#FF | Input is not connected |

### General rules

- If a given input causes a signal change at the output, the signal status of the input with the highest priority (worst signal status) is set at the output, since each input can influence the output:
  - Logical AND operation (logic = 1):
    If the output value is 1, it has the signal status with the highest priority of all interconnected input signals.
  - Logical OR operation (logic = 0):
    If the output value is 0, it has the signal status with the highest priority of all interconnected input signals.
- If several inputs have the same priority and the output therefore cannot be changed, the signal status of the input with the lowest priority (highest signal status) is set at the output, since the output is uniquely determined by the signal with the best signal status:
  - Logical AND operation (logic = 1):
    If the output value is 0, it has the signal status with the lowest priority of all interconnected input signals, which have a value of 0.

    **Note**

    With `Feature.Bit23` = 1 (evaluation signal status), input signals with a bad signal status (`16#28` or `16#00`) are processed with the value 0, regardless of their actual value

  - Logical OR operation (logic = 0):
    If the output value is 1, it has the signal status with the lowest priority of all interconnected input signals, which have a value of 1.
- If no inputs are interconnected, the signal status of the output is set to simulation (`16#60`).

### Display of the signal status in the faceplate and block icon for interlocking blocks

The signal status is displayed for each individual parameter (except for the analog values) in the faceplate next to the process values.

If you bypass a signal, it is displayed in the faceplate of the interlocking block next to the button for excluding, as well as in the block icon as follows.




**Note**

If an excluded interlock signal can become switchover-relevant, the status of the output Out.ST is set to simulation 16#60, as long as it is not forced by other interlock inputs to Bad, device-related 16#00 or Bad, process-related 16#28.

An excluded interlock input has no influence on the status when the Separate evaluation for excluded and simulated interlock signals (Page 162) function is enabled.

The currently valid status for the output signal is also displayed in the faceplate.

### 2.1.5.8 Forming and outputting the signal status for mathematical blocks

#### Forming the signal status for mathematical blocks

For more general information on forming the status signal, refer to the section: Forming and outputting signal status for blocks (Page 116).

The status for the result of the output is formed within mathematical blocks from all process inputs involved in the calculation, according to the following table (highest priority is 0)

| Signal status icon | Priority | Value | Meaning |
|---|---|---|---|
|  | 0 | 16#00 | Bad, device related |
|  | 1 | 16#28 | Bad, process related |
|  | 2 | 16#60 | Manipulated value (for example, substitute value, simulation, last valid value) |
|  | 3 | 16#68 | Uncertain, device related |
|  | 4 | 16#78 | Uncertain, process related |
|  | 5 | 16#A4 | Maintenance request |
|  | 6 | 16#80 | Good |

The group status is set to 16#68 (uncertain, device-related), with an undefined signal status at a control input that is involved in forming the group status.

A 16#FF signal status at a control input is not used for the calculation of the group status. If all relevant control inputs are 16#FF, the group status in the search for the worst group status is 16#80 and 16#00 in the search for the best group status.

If only one process input is decisive for calculating the output value, the status of the process input is transferred to the status of the output.

**Note**

**Special notes for the Integral mathematical block**

Due to its application area (time integral, I-component for the configuration of a controller), the Integral block generates the signal status like the technological blocks.

#### Special notes for the mathematical blocks Addxx, Mulxx, Div02 and Sub02

If the result of the mathematic operation is a floating-point number that cannot be displayed, the result of the status is set to 16#28. Floating-point numbers that cannot be displayed are labeled in the CFC with #+Inf (+ infinite), #-Inf (- infinite) or with #NaN (not a number).

### 2.1.5.9 Forming and outputting the signal status for PCS 7 channel blocks

#### Forming the signal status for PCS 7 channel blocks

For more general information on forming the status signal, refer to the section: Forming and outputting signal status for blocks (Page 116).

The signal status for PCS 7 channel blocks can assume the following values:

| Signal status icon | Value | Meaning |
|---|---|---|
|  | 16#80 | Good |
|  | 16#78 | Uncertain, process related:<br>Limitation of input parameter `PV_In` is active (analog output channel blocks only) |
|  | 16#60 | Manipulated value (for example, substitute value, simulation, last valid value) |
|  | 16#00 | Bad, device related (value not valid) |
|  | 16#A4 | Maintenance required, maintenance demanded |
|  | 16#68 | Uncertain, device related |

### 2.1.5.10 Forming and outputting the signal status for channel blocks for field devices

#### Forming the signal status for channel blocks (field devices)

For more general information on forming the status signal, refer to the section: Forming and outputting signal status for blocks (Page 116).

The signal status of channel blocks for field devices can assume the following values:

| Signal status icon | Value | Meaning |
|---|---|---|
|  | 16#80 | Good |
|  | 16#78 | Uncertain, process related |
|  | 16#68 | Uncertain, device related |
|  | 16#60 | Manipulated value (for example, substitute value, simulation, last valid value) |
|  | 16#28 | Bad, process related |

| Signal status icon | Value | Meaning |
|---|---|---|
|  | 16#00 | Bad, device related (value not valid) |
|  | 16#A4 | Maintenance required, maintenance demanded |

## 2.1.6 Error handling

### 2.1.6.1 Error handling

#### Error handling

The channel and technologic blocks feature error handling routines. A distinction must be made between the following areas:

- Error numbers
- External process control error (CSF)
- Process-specific errors
- Invalid signal states
- Mode switchover error
- Errors in channel blocks

#### Error numbers

Most blocks have an output parameter `ErrorNum` that can be used to output internal error states of the block as error numbers.

With some blocks, input parameters are checked for permissible values. They are therefore only used to prevent the output value from remaining invalid when the input value is once again in the valid range. If an invalid value is detected, and the corresponding output value is held at the last displayed value instead of an invalid value being displayed. If blocks do not have this check, an invalid value can appear at the output. However, a valid value is displayed again at the output as soon as the input values of the block have changed correspondingly.

Any value set over an interconnection or as a result of a parameter assignment that is outside the range of values (e.g. "Not a Number") is not processed by the block algorithm. The last valid value is processed instead.

In addition to the errors stated above, a limit violation is also signaled for example. Each error number is assigned to a specific error.

If there is more than one error, all error numbers have the same priority. The routine always displays the error number of the error most recently detected in a block cycle.

### External process control error (CSF)

An external process control error always lies outside the process - it exists in the form of device or other hardware faults. If, for example, a run-time error occurs at a valve, there is an error or fault in the pneumatic system.

A process control error is output if an external error is set at the input `CSF`. You can enable this output function, for example, by interconnecting output `Bad` of the channel block with input `CSF` of the technologic block.

The error message "$$BlockComment$$ External error occurred" is output at `CSF = 1`.

This state is visualized in the group display by an "S" character in the faceplate overview and in the block icon.

With motor, valve and dosing blocks, there is also the possibility that the block switches to error processing. `Feature.Bit18` Activating error state for external process control error CSF (Page 162) must be set for this. The device goes into error processing and moves the drive to the neutral position or switches the dosing operation to off for a doser. If the dosing operation is already finished, it remains in the completed state. Resetting the error is described in Resetting the block in case of interlocks or errors (Page 47).

### External error (FaultExt)

The `FaultExt` input can be used to pass an external error without the generation of a message. The device goes into error processing and moves the drive to the neutral position or switches the dosing operation to off for a doser. If the dosing operation is already finished, it remains in the completed state. Separate messages can be generated by interconnecting the external error to `ExtMsgx` messages that can be freely selected. Resetting the error is described in Resetting the block in case of interlocks or errors (Page 47).

### Process-specific errors

Process-specific errors can have the following causes:

- Runtime monitoring: If the feedback signals do not match the control settings after a selected time has expired, a process-related error is output.
- Feedback monitoring: Refer to the Monitoring the feedbacks (Page 104) section for more on this.

If the block algorithm detects a monitoring error while monitoring is enabled, the corresponding output parameter is set to 1 in the block. The "$$BlockComment$$ Feedback error xxx" error message is also output, where xxx, for example, stands for the valve.

This state is visualized in the group display by an "S" character in the faceplate overview and in the block icon.

The block must be reset after the monitoring error was cleared and if automatic mode is set.

### Invalid input signals

This error is output if inconsistencies are detected between associated I/Os. The close and open commands cannot be output simultaneously to the valve, for example.

If the block algorithm detects an invalid combination of input signals, an error number (`ErrorNum`) is output that depends on the block type.

In the case of motors, valves and batches, the faceplates' standard view will additionally output the text "invalid signal".

### Mode switchover error

This error is reported if you change the mode of the block from:

- Manual to automatic mode or
- Local mode to automatic mode

and the previous and target state are **inconsistent** (bumpless switchover). You can only change the block mode if the subsequent state corresponds with the previous state.

Bumpless switchover can be activated / deactivated using the `Feature` connection on the Enabling bumpless switchover to automatic mode for valves, motors, and dosers (Page 184) or Disabling bumpless switchover to automatic mode for controllers (Page 183) bit.

Bumpless switchover from local to automatic mode is undertaken using the `LocalSetting` parameter, as described in section Local mode (Page 86).

In the standard view of the faceplate, the text "Changeover error" is displayed in the event of an unwanted switchover with bumps.

The block retains local mode if the operator changes the mode from local to automatic and the error mentioned above occurs. The block changes to manual mode if the mode is changed from local to automatic over interconnected inputs and the error mentioned above occurs.

### Errors in channel blocks

The following errors may be displayed by the channel blocks:

- Channel error
- Device or module fault
- Higher-level error
- Invalid measuring range

## 2.1.6.2 Outputting group errors

### Outputting group errors

The `GrpErr` output parameter assembles the faults of a block and makes them available to you. A group error is compiled from the following error information:

- Feedback errors (static or dynamic feedback monitoring)
- External error (CSF and `FaultExt`)
- Motor protection (only for motors)
- Module errors (only for hardware controllers)

- I/O access errors (only for hardware controllers)
- Parameter assignment errors (only for hardware controllers)

For information on how the signal is formed for the group error at the `GrpErr` output parameter, refer to the corresponding block descriptions in the "Functions" chapter.

**Note**

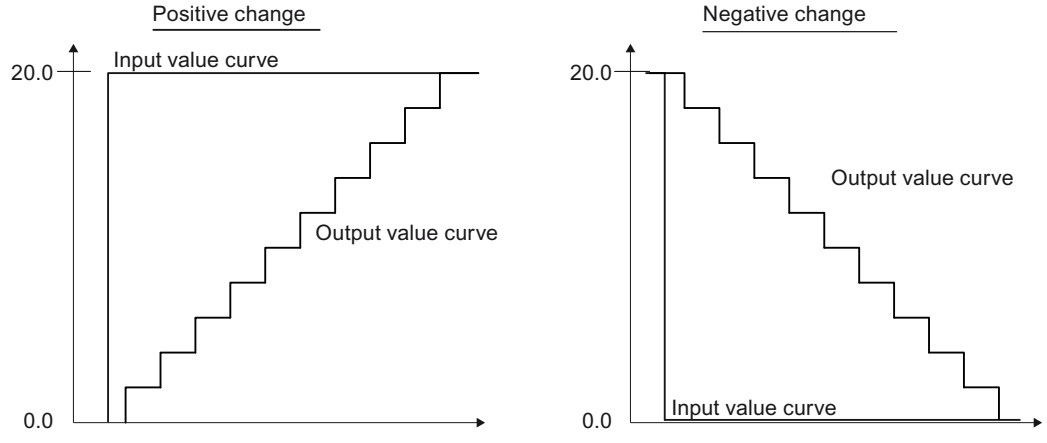
The signal status of the individual signals is not taken into consideration for forming the group error. The outputs therefore always have the status, `16#80`.

## 2.1.7 Ramp function

### 2.1.7.1 Using setpoint ramp

#### Using setpoint ramp

Starting at the current internal setpoint, the setpoint can be set to a target setpoint value over a ramp-shaped function. In the faceplate, you can start the function in ramp view (`SP_RmpOn = 1`).




Use the `SP_RmpModTime` input parameter or the ramp view of the faceplate to specify whether the setpoint ramp is defined by time or by gradients:

- If you select time `(SP_RmpModTime = 1)`: The ramp of the setpoint is calculated automatically by the block so that after the ramp has started `(SP_RmpOn = 1)`, the setpoint will reach the target setpoint (`SP_RmpTarget`) after the selected time `(SP_RmpTime)`.
- If you select gradients `(SP_RmpModTime = 0)`: The inclination of the ramp matches the selected rates of change `SP_UpRaLim` (positive) or `SP_DnRaLim` (negative).

Once the setpoint has reached the target setpoint, the function is terminated automatically (SP_RmpOn = 0). The ramp trip can be prematurely aborted in the faceplate by setting SP_RmpOn = 0.

**Requirements for using a setpoint ramp**

| Block | Setpoint | Gradient limit | Operating mode |
|---|---|---|---|
| Controller | Internal | Off | Automatic |
| OpAnL | Internal | Off | |
| MotSpdCL | Internal | Off | |

If the requirements are not fulfilled during the ramp trip, the ramp trip is automatically canceled.

**Note**

**Special note for the MotSpdCL block**

In the case of an interlock, a monitoring error, motor protection or rapid stop, the motor is switched off and the internal ramp setpoint is reset to the starting setpoint of the ramp trip.

### 2.1.7.2 Gradient limit of the setpoint

**Gradient limit of the setpoint**

The gradient limit is activated via the SP_RateOn = 1 input parameter.

The values are set at the SP_UpRaLim and SP_DnRaLim parameters depending on the TimeFactor.

- TimeFactor = 0: Unit of the gradient limiting is Unit/Second, Unit of ramp time is Second
- TimeFactor = 1: Unit of the gradient limiting is Unit/Minute, Unit of ramp time is Minute
- TimeFactor = 2: Unit of the gradient limiting is Unit/Hour, Unit of ramp time is Hour
- SP_UpRaLim sets the gradient high limit
- SP_DnRaLim sets the gradient low limit

**Note**

ParametersSP_UpRaLim and SP_DnRaLim are always evaluated according to their magnitude.

With the Feature Bit Gradient limitation with time duration (Page 197), you can also use the parameters SP_RmpModTime and SP_RmpTime in the gradient limit function. If this Feature Bit = 1 and SP_RmpModTime = 1 is parameterized, the ramp rate is calculated with a change of setpoint such that the new value of the setpoint will reach after the time of SP_RmpTime.

### External setpoint

If the external setpoint (SP_ExtAct.Value = 1) is switched on, the linkable values SP_LiUpRaLim, SP_LiDnRaLim, SP_LiRmpTime, of SP_UpRaLim, SP_DnRaLim, SP_RmpTime respectively will be used. The values will be active only if they are connected (SP_LiX.ST <> 16#FF).

### Displaying active limitation

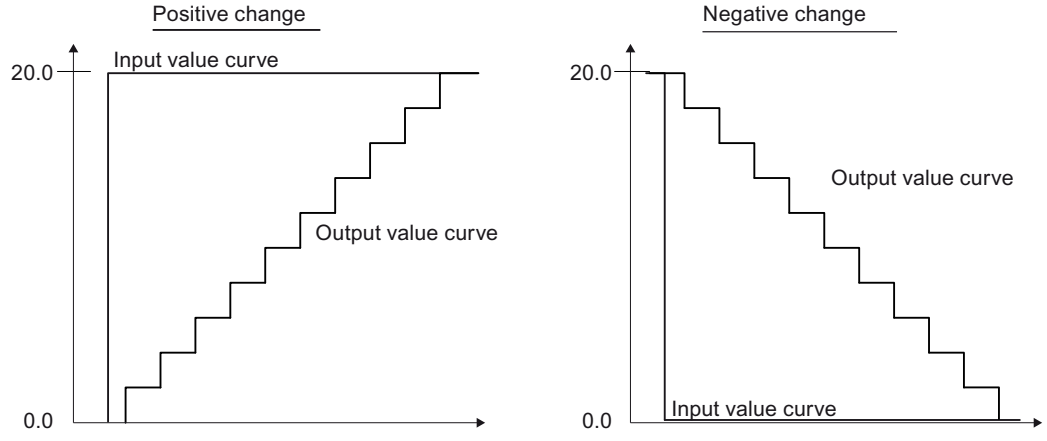
A gradient limit is indicated at the following output parameters:

- SP_UpRaAct = 1: Gradient has a high limit
- SP_DnRaAct = 1: Gradient has a low limit

## 2.1.7.3 Using a manipulated variable ramp

### Using a manipulated variable ramp

Starting at the current internal manipulated variable, the manipulated variable can be brought to a target value in the form of a ramp. In the faceplate, you can start the function in ramp view (MV_RmpOn = 1)).




Use the MV_RmpModTime input parameter or the ramp view of the faceplate to specify whether the manipulated variable ramp is defined by time or by gradients:

- If you select time (MV_RmpModTime = 1): The gradients of the manipulated variable are calculated automatically by the block so that after the ramp has started (MV_RmpOn = 1), the manipulated variable will reach the target value (MV_RmpTarget) after the configured time (MV_RmpTime).
- Specification with gradients (MV_RmpModTime = 0): The ramp slope matches the configured rates of change MV_UpRaLim (positive) or MV_DnRaLim (negative).

Once the manipulated variable has reached the target value, the function is terminated automatically (MV_RmpOn = 0). The ramp trend can be prematurely aborted in the faceplate by setting MV_RmpOn = 0.

**Requirements for using a manipulated value ramp:**

| Block | Manipulated variable | Gradient limit | Operating mode |
|---|---|---|---|
| VlvAnL without auxiliary valve | Internal | Off | Manual |
| VlvAnL with auxiliary valve | Internal | Off | |

If the requirements are not fulfilled during the ramp trip, the ramp trip is automatically canceled.

### 2.1.7.4 Gradient limiting of the manipulated variable

#### Gradient limiting of the manipulated variable

The gradient limit is activated via the `MV_RateOn = 1` input parameter.

The values are set at the `MV_UpRaLim` and `MV_DnRaLim` parameters depending on the `TimeFactor`.

- `TimeFactor = 0`: Unit of the gradient limiting is Unit/Second, Unit of ramp time is Second
- `TimeFactor = 1`: Unit of the gradient limiting is Unit/Minute, Unit of ramp time is Minute
- `TimeFactor = 2`: Unit of the gradient limiting is Unit/Hour, Unit of ramp time is Hour
- `MV_UpRaLim` sets the gradient high limit
- `MV_DnRaLim` sets the gradient low limit

**Note**

Parameters `MV_UpRaLim` and `MV_DnRaLim` are always evaluated according to their magnitude.

With the `Feature` Bit Gradient limitation with time duration (Page 197), you can also use the parameters `MV_RmpModTime` and `MV_RmpTime` in the gradient limit function. If this `Feature` Bit = 1 and `MV_RmpModTime` = 1 is parameterized, the ramp rate is calculated with a change of manipulated variable such that the new value of the manipulated variable will reach after the time of `MV_RmpTime`.

#### External maniputaed variable

If the external maniputaed variable (`MV_ExtAct.Value` = 1) is switched on, the linkable values `MV_LiUpRaLim`, `MV_LiDnRaLim`, `MV_LiRmpTime`, of `MV_UpRaLim`, `MV_DnRaLim`, `MV_RmpTime` respectively will be used. The values will be active only if they are connected (`MV_LiX.ST` <> `16#FF`).

#### Displaying active limitation

A gradient limit is indicated at the following output parameters:

- `MV_UpRaAct = 1`: Gradient has an high limit
- `MV_DnRaAct = 1`: Gradient has a low limit

## 2.1.8 Internal/external setting

### 2.1.8.1 Applying the dynamically activated dead band during the PV settling time

#### `Feature` bit

Number of the `Feature` bit: 30.

#### Dynamic deactivation of the dead band by the `PV` settling time

You can use this `Feature` bit to improve the `PV` settling time in the range of the dead band.

The deadband is deactivated until the `PV` settles more in the center than at the edges of the dead band. The probability that the controlled variable is within the dead zone time is increased, and further control actions are no longer necessary. This leads to reduced wear and less energy consumption.

**Bit = 0**: Dead band width is constant

**Bit = 1:** The dead band is deactivated for the `PV` settling time. When the `PV` settling time is over, the dead zone gets the value of the `DeadBand` parameter.

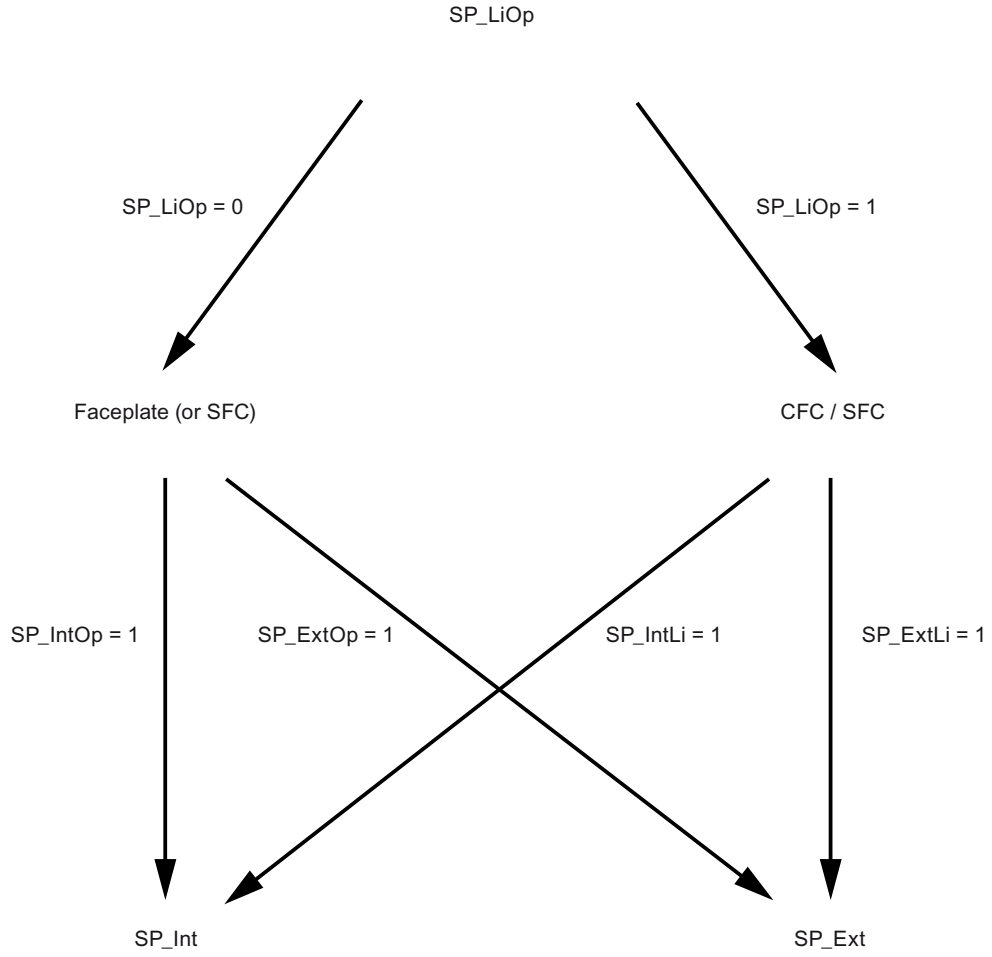
**Note**

If `Feature.Bit30` = 1 is set, the ConPerMon block has to be connected with the controller. The blocks PIDConL or PIDConR monitor this by checking the status signal of `MeanMV.ST`. In case of `Feature.Bit30` = 1 and `MeanMV.ST` = 16#78 the dynamic deactivation of deadband is not possible and the output `ErrorNum` will be set to 52.

### 2.1.8.2 Setpoint specification - internal/external

#### Setpoint specification internal & external

Some blocks have a function that allows setpoints to be specified. This specification is carried out either by means of a CFC/SFC program or by means of the faceplate (operator). With doser blocks and frequency converters, the operator can specify the internal setpoint value (`SP_Int`) or a higher-level open-loop control will specify an external setpoint value (`SP_Ext`). In principle, the blocks operate according to the same scheme:




First you define whether the setpoint specification is to be carried out by means of a CFC/SFC program or by means of the faceplate. In the next step you specify whether the internal or the external setpoint is to be used.

**Setpoint specification by means of faceplate or interconnection**

With the SP_LiOp parameter, you define whether the setpoint will be set by a CFC/SFC program or using the faceplate.

- Parameterize Sp_LiOp with 0 so that the setpoint specification is carried out by means of the faceplate.
- Parameterize SP_LiOp with 1 so that the setpoint specification is carried out by means of a CFC / SFC program.

**Setpoint specification internal & external**

You have to set the corresponding parameters depending on how the setpoint specification is to be carried out.

If the setpoint is set in the faceplate (SP_LiOp = 0), you have to set the parameter:

- SP_IntOp = 1 in order to achieve an internal setpoint specification by means of the faceplate.
- SP_ExtOp = 1 to have an external setpoint set in the faceplate.

If both signals are set, SP_IntOp = 1 has priority.

If the setpoint is set by a CFC / SFC program (SP_LiOp =1), you have to set the parameter:

1. SP_IntLi = 1 to have an internal setpoint set by a CFC / SFC program.
2. SP_ExtLi = 1 in order to achieve an external setpoint specification by means of a CFC / SFC program.

**Note**

For PIDConL, PIDStepL, FmCont, FmTemp: If both signals are set, SP_IntLi = 1 has priority.

### Bumpless switchover from external to internal setpoint

The parameter SP_TrkExt = 1 is used so that the internal setpoint tracks the external setpoint to achieve a Bumpless switchover from the external to the internal setpoint. This allows unwanted jumps at the output parameter to be avoided.

## 2.1.8.3 Manipulated variable specification - internal/external

### Manipulated variable specification internal and external

The VlvAnl block provides a function for specifying manipulated variables. This specification is carried out either using a CFC/SFC program or using a faceplate (operator).

If a auxiliary valve is used for operation, it is possible to switch between internal and external in the both manual and automatic mode.

If no auxiliary valve is used for operation, the external manipulated variable is used for automatic mode and the internal manipulated variable is used for manual mode. It is not possible to switch between internal and external.

**Note**

If no auxiliary valve is used and either the "Open" or "Close" command is active, a new manipulated variable (internal or external) only takes effect after the change. In automatic mode, the manipulated variable is ignored as long as the "Open" or "Close" command is pending.

### Manipulated variable specification using a faceplate or interconnection

You can use the `MV_LiOp` parameter to determine if the manipulated variable should be set by a CFC/SFC program or via the faceplate.

- Set `MV_LiOp` to 0 for manipulated variable specification to be performed with the faceplate.
- Set `MV_LiOp` to 1 for manipulated variable specification to be performed by a CFC/SFC program.

### Manipulated variable specification internal and external

You need to set the corresponding parameters depending on the selected method for manipulated variable specification.

If the manipulated variable is to specified via the faceplate `MV_LiOp = 0`, you have to set the parameter:

- `MV_IntOp = 1` in order to achieve internal manipulated variable specification via the faceplate.
- `MV_ExtOp = 1` in order to achieve external manipulated variable specification via the faceplate.

If both signals are set, the most recently active state is retained.

If the manipulated variable is set by a CFC/SFC program (`MV_LiOp = 1`), you have to set the parameter:

- `MV_IntLi = 1` to have an internal manipulated variable set by a CFC/SFC program.
- `MV_ExtLi = 1` in order to achieve an external manipulated variable specification by a CFC/SFC program.

If both signals are set, `MV_IntLi = 1` has priority.

**Note**

It is only possible to switch between internal and external when operating with a auxiliary valve.

#### Bumpless switchover of the manipulated variable from external to internal

The parameter `SP_TrkExt = 1` is used so that the internal setpoint tracks the external setpoint to achieve a Bumpless switchover from the external to the internal setpoint. This allows unwanted jumps at the output parameter to be avoided.

#### Forming the manipulated variable externally

With external manipulated variable specification, the manipulated variable is restricted to the `MV_HiLim` and `MV_LoLim` limits and sent to the `MV_ExtOut` output.

## 2.1.9 Configurable response using the Feature I/O

### 2.1.9.1 Configurable functions using the Feature I/O

#### Configurable functions using the `Feature` I/O

Some blocks have an input called `Feature`. This input can be used to influence the way in which the block works.

The `Feature` bits are assigned in the following order:

| Bit number | Meaning | Block |
|---|---|---|
| 0 | Set startup characteristics (Page 148) | AV, Average, CountOh, CountScL, DeadTime, Derivative, DoseL, FmCont, FmTemp, Integral, Lag, MeanTime, ModPreCon, MonAnL, MonAnS, MonDi08, MonDiL, MonDiS, MotL, MotS, MotRevL, MotSpdCL, MotSpdL, OpAnL, Pcs7AnOu, Pcs7DiOu, PIDConL, PIDConR, PIDStepL, RateLim, Ratio, ShrdResS, TotalL, Vlv2WayL, VlvAnL, VlvMotL, VlvL, VlvS |
| 1 | Reaction to the out of service mode (Page 188) | ConPerMon, CountOh, CountScL, DoseL, Event, EventNck, EventTs, FmCont, ModPreCont, MonAnL, MonAnS, MonDi08, MonDiL, MonDiS, MotL, MotS, MotRevL, MotRevSpdL, OpAnL, OpDi01, OpDi03, OpTrig, PIDConL, PIDConR, PIDStepL, Ratio, SelA16In, TotalL, Vlv2Way, VlvAnL, VlvL, VlvS, VlvMotL |
| | Changes signal status of outputs in OOS (Page 208) | MotL, MotS, MotRevL, MotSpdL, MotSpdCL, VlvS, VlvL, Vlv2WayL, VlvMotL, VlvAnL, VlvPosL |

| Bit number | Meaning | Block |
|---|---|---|
| 2 | Resetting the commands for changing the mode (Page 172) | DoseL, FmCont, FmTemp, ModPreCon, MotL, MotS, MotRevL, MotSpdCL, MotSpdL, PIDConL, PIDConR, PIDStepL, Vlv2WayL, VlvAnL, VlvL, VlvS, VlvMotL, SelA16In |
| | Separate evaluation for excluded and simulated interlock signals (Page 162) | Intlk02, Intlk04, Intlk08, Intlk16, OpDi01, OpDi03 |
| 3 | Enabling resetting of commands for the control settings (Page 172) | DoseL, MotL, MotS, MotRevl, MptSpdCL, MotSpdL, Vlv2WayL, VlvAnL, VlvL, VlvMotL |
| | Control priority in the event of invalid or not plausible input commands (Page 186) | DoseL, MotL, MotS, MotRevL, MotSpdCL, MotSpdL, Vlv2WayL, VlvL, VlvS, VlvMotL, VlvAnL, VlvPosL, |
| | Changes signal status of outputs in OOS (Page 208) | SelA16In, SelAn16L |
| | Disable limit act output when limit value message is disabled (Page 209) | MonAnL, MonAnS, PIDConS, AV, CountOh, CountScL, TotalL, PumpMon |
| | Bypass over both connection BypLix OR Inx (Page 212) | Intlk02, Intlk04, Intlk08, Intlk16 |
| 4 | Setting switch or button mode (Page 177) | DoseL, FMCont, FMTemp, ModPreCon, MotL, MotRevL, MotSpdCL, MotSpdL, PIDConL, PIDStepL, Vlv2WayL, VlvAnL, VlvL, VlvMotL, SelA16In |
| | Setting switch or button mode for local commands (Page 195) | MotL, MotRevL, MotSpdCL, MotSpdL, VlvL, Vlv2WayL, VlvMotL, VlvPosL |
| | Changes signal status of outputs in OOS (Page 208) | CntOhSc, CountScL, CountOh, TotalL |

| Bit number | Meaning | Block |
|---|---|---|
| 5 | Specifying the dosing type (Page 157) | DoseL |
| | Control via auxiliary valve (Page 182) | VlvAnL |
| | Alarm setpoint difference (Page 182) | MotSpdCL |
| | Specifying switching mode (Page 178) | MotSpdL |
| | Use the last value following a complete download as the current value during startup of the block (Page 165) | TotalL |
| | Display only input values that are interconnected in the faceplate (Page 168) | SelA16In |
| | Activate OS_Perm bits (Page 168) | Intlk02, Intlk04, Intlk08, Intlk16 |
| | Limit output Out (Page 154) | Polygon |
| | Setting the scaling for the process values (Page 159) | FbDrive |
| | Activate LowCutOff (Page 166) | Pcs7AnIn |
| | Evaluation of the signal status of the interlock signals (Page 153) | OpDi01, OpDi03 |
| | | DoseL, MotL,MotRevL,MotS, MotSpdCL,MotSpdL,Vlv2WayL,VlvAnL,VlvL,VlvS, VlvMotL |
| | Retain last output value in case of bad input signal status (Page 198) | MeanTime |
| | Changes signal status of outputs in OOS (Page 208) | PIDConS, PIDConR, PIDConL, PIDStepL, FmTemp, FmCont, MPC 10x10, KalFit, ModPreCon, TimeTrig, AV, MonAnL, MonAnS, MonDiL, MonDi08, MonDiS |
| | Timestamp calculation (Page 210) | Pcs7DiIT |
| | Reversal of control direction for fail-open valves (Page 211) | Pcs7AnOu |
| 6 | Resetting the dosing quantity when dosing starts (Page 173) | DoseL |
| | Block as summing unit or integrator (Page 154) | TotalL |
| | Disabling opening and closing (Page 169) | VlvAnL |
| | Substitute value switch in the event of an error (Page 160) | Polygon |
| | Failure handling (Page 148) | FbDrive |
| | Vibrate after torque monitoring (Page 190) | VlvPosL |
| | Ramp rate calculation (Page 191) | PIDConL, PIDConR, PIDStepL, MotSpdCL, OpAnL, FMTemp, FMCont |
| | External/internal selection specification (Page 194) | SelA16In |
| | Operator can change the setpoint via faceplate also in the "Local" mode (Page 197) | MotSpdCL |
| | Scalable measuring range (Page 203) | Pcs7AnIn |
| | Changes signal status of outputs in OOS (Page 208) | DoseL |
| | External Input to enable or Disable (Page 211) | TimeTrig |

| Bit number | Meaning | Block |
|---|---|---|
| 7 | Enabling direct changeover between forward and reverse (Page 156) | MotRevL, MotSpdCL |
| | Summing characteristics continuous or triggered (Page 184) | TotalL |
| | Activating calculation of the flow rate for dosing by scale (Page 155) | DoseL |
| | Ramp rate calculation (Page 191) | VlvAnL |
| | Analog input 1 is reserved for the operator (Page 197) | SelA16In |
| | Define the setpoint after stop and start of the motor (Page 198) | MotSpdCL |
| | 2 decimal points (scalable measuring range) (Page 203) | Pcs7AnIn |
| | Behavior in the case of Bus failure (Page 208) | FbDrive, FbSwtMMS |
| | Switch over to internal time enable or Disable (Page 211) | TimeTrig |
| | Use connectable inputs OpenLi, StopLi, CloseLi with CtrlLiOp (Page 212) | PIDStepL |
| 8 | Unit for the rate of change (Page 158) | RateLim |
| | Sealing the valve (Page 187) | DoseL |
| | Reporting with BATCH parameters (Page 167) | Event, EventTS |
| | Sealing the valve (Page 187) | VlvMotL |
| | Switch to substitute value (Page 159) | FlowCorr |
| | Separate delay times for each alarm (Page 181) | MonAnL, PIDConL, PIDConR |
| | Forcing commands in the Local mode (Page 198) | MotL, MotRevL, MotSpdCL, MotSpdL, VlvL, Vlv2WayL, VlvMotL, VlvPosL, DoseL |
| | Inverter enable (Page 201) | MotSpdCL |
| | Delete history of MV and DV moves during restart (Page 201) | ModPreCon, MPC10x10 |
| | Status output calculation (Page 202) | Integral |
| | Variants of CPI calculation (Page 202) | ConPerMon |
| | 3 decimal points (scalable measuring range) (Page 204) | Pcs7AnIn |
| | AS time enable or Disable (Page 212) | TimeTrig |
| 9 | Resetting via input signals in the event of interlocking (Protection) or errors (Page 174) | DoseL, MotL, MotS, MotRevL, MotSpdCL, MotSpdL, Vlv2WayL, VlvAnL, VlvL, VlvS, VlvMotL |
| | Substitution value is active if the block is in bypass (Page 199) | MonAnL, MonDiL, PIDConL, PIDConR, PIDStepL, TotalL |
| | Enable external message (Page 200) | EventTs, Event16Ts |
| | Selection of metadata and diagnostic blocks (Page 205) | FbSwtMMs, FbDrive |
| | Button mode with triggered local control commands (Page 209) | MotL, MotSpdCL, MotRevL, MotSpdL, Vlv2WayL, VlvL, VlvPosL, VlvMotL |

| Bit number | Meaning | Block |
|---|---|---|
| 10 | Exiting local mode (Page 189) | DoseL, MotL, MotS, MotRevL, MotSpdCL, MotSpdL, Vlv2WayL, VlvAnL, VlvL, VlvS, VlvMotL |
| | Condition monitoring information at MOD_Blocks (Page 155) | FbAnIn, FbAnOu, FbDiIn, FbDiOu, FbAnTot, Pcs7AnIn, Pcs7AnOu, Pcs7DiIn, Pcs7DiOu, Pcs7Cnt1, Pcs7Cnt2, Pcs7Cnt3, FmCont, FmTemp |
| | Condition monitoring information at the channel blocks (Page 155) | FbAnIn, FbAnOu, FbAnTot, FbDiIn, FbDiOu, FbDrive, FbEnMe, FbSwtMMS, Pcs7AnIn, Pcs7AnOu, Pcs7Cnt1, Pcs7Cnt2, Pcs7Cnt3, Pcs7DiIn, Pcs7DiIT, Pcs7DiOu, FmCont, FmTemp, Pcs7AITC, Pcs7DiCnt |
| | Considering bad quality of automatic commands or external values (Page 200) | MotL, MotS, MotRevL, MotSpdCL, MotSpdL , Vlv2WayL, VlvL, VlvS, VlvMotL, VlvPosL, VlvAnL, DoseL, OpAnL, OpAnS, OpDi01, OpDi03, OpTrig |
| 11 | Stopping dosing at a flow alarm (Page 147) | DoseL |
| | Activating the run time of feedback signals (Page 164) | MotL, MotS, MotRevL, MotSpdL, MotSpdCL, VlvL, Vlv2WayL, VlvMotL, VlvAnL |
| | Gradient limitation with time duration (Page 197) | PIDConL, PIDConR, PIDStepL |
| | Suppress MsgLock and “Out of service” mode for a connected message block (Page 202) | DoseL, FmCont, FmTemp, PIDConL, PIDConR, PIDStepL, MonDi08, MonAnL, MonDiL, MotL, MotSpdCL, MotRevL, MotSpdL, VlvL, VlvMotL, Vlv2WayL, VlvAnL, VlvPosL |
| | Understeering / underflow for measuring range 0...10V (Page 209) | Pcs7AnIn |
| 12 | Automatic post dosing for underdosing in automatic mode (Page 153) | DoseL |
| | Control zone with specified I component (Page 171) | PIDConL |
| | Motor feedback is not available (Page 168) | VlvMotL, VlvPosL |
| | Position feedback signals are active (Page 179) | Vlv2WayL |
| | Gradient limitation with time duration (Page 197) | MotSpdCL, VlvAnL, OpAnL |
| | Disable limit act output when limit value message is disabled (Page 209) | PIDConL, PIDConR, PIDStepL, ConPerMon, DoseL, VlvPosL, MotSpdCL, FmCont, FmTemp, VlvAnL |
| | Selecting the Feature Bit when Namur Off = 1 (Page 212) | Pcs7AnIn |
| 13 | Creep rate is always detected in the dosing quantity (Page 179) | DoseL |
| | Control zone with frozen I component (Page 171) | PIDConL |
| | Separate monitoring time for stopping the motor (Page 180) | MotL, MotRevL, MotSpdCL, MotSpdL, VlvMotL |
| | Resetting the Pending Error in Local Mode 1 and 3 (Page 210) | MotL, MotRevL, MotSpdcL, MotSpdL, VlvMotL, VlvPosL, Vlv2WayL, VlvL. |

| Bit number | Meaning | Block |
|---|---|---|
| 14 | Enabling rapid stop via faceplate (Page 179) | MotL, MotRevL, MotSpdCL, MotSpdL, VlvMotL |
| | External control deviation (Page 161) | PIDConL |
| | Scale dosing can be started only if StandStill = 1 (Page 201) | DoseL |
| | With PID initialization if mode SP intern extern changes (Page 207) | PIDConR |
| 15 | Neutral position manipulated variable takes effect with "out of service" operating mode (Page 177) | ModPreCon, PIDConL, PIDConR, PIDStepL, VlvAnL |
| | Frequency converter with separate device feed (Page 162) | MotSpdL |
| | Frequency converter with separate device feed (Page 162) | DoseL |
| | Motor stop in end position depends only on the corresponding feedback signal (Page 199) | VlvMotL |
| 16 | Neutral position manipulated variable takes effect at startup (Page 176) | ModPreCon, PIDConL, PIDConR, PIDStepL, VlvAnL |
| | Process value with separate scale range (Page 173) | OpAnL |
| | Setpoint specification with own scale and unit of the parameter (Page 182) | MotSpdCL |
| | Separate interlock for each direction, position, or speed (Page 180) | MotRevL, VlvAnL, MotSpdL |
| | Limits disabled in faceplate (Page 206) | DoseL |
| 17 | Enabling bumpless switchover to automatic mode for valves, motors, and dosers (Page 184) | DoseL, MotL, MotRevL, MotSpdCL, MotSpdL, Vlv2WayL, VlvAnL, VlvL, VlvMotL |
| | With accelerated return of the integral action from the manipulated variable limit (Page 193) | PIDConR |
| | Resynchronization after startup (Page 213) | MotL, MotRevL, MotSpdL, MotSpdCL, VlvL, VlvMotL, VlvAnL, VlvPosL, OpDi01, OpDi03, OpAnL |
| 18 | Disabling bumpless switchover to automatic mode for controllers (Page 183) | PIDConL, PIDConR, PIDStepL |
| | Activating error state for external process control error CSF (Page 162) | DoseL, MotL, MotS, MotRevL, MotSpdCL, MotSpdL, Vlv2WayL, VlvAnL, VlvL, VlvS, VlvMotL |
| | Resynchronization after out of service or after simulation (Page 213) | MotL, MotRevL, MotSpdL, MotSpdCL, VlvL, VlvMotL, VlvAnL, VlvPosL, OpDi01, OpDi03, OpAnL |
| 19 | Enabling program mode (Page 170) | PIDConR |
| | Reset even with locked state (Page 176) | MotL, MotS, MotRevL, MotSpdCL, MotSpdL, VlvMotL |
| | Resynchronization after CSF (Page 214) | MotL, MotRevL, MotSpdL, MotSpdCL, VlvL, VlvMotL, VlvAnL, VlvPosL |

| Bit number | Meaning | Block |
|---|---|---|
| 20 | Enabling bumpless change to the proportional gain, derivative time and amplification of the differentiator (Page 183) | PIDConR |
| | Disable calculation of impulse control in LocalSetting 2 and 4 (Page 200) | MotL, MotRevL, MotSpdCL, MotSpdL, VlvL, VlvMotL, VlvPosL, Vlv2WayL |
| | Resynchronization with StateTrkOn input (Page 214) | MotL, MotRevL, MotSpdL, MotSpdCL, VlvL, VlvMotL, VlvAnL, VlvPosL, OpDi01, OpDi03, OpAnL |
| 21 | Enable bumpless switchover to "Automatic" mode for operator only (Page 183) | MotL, MotRevL, MotSpdCL, MotSpdL, VlvMotL, VlvL, Vlv2WayV, VlvAnl, DoseL |
| | Switching operator controls for external setpoint to visible (Page 154) | PIDConR |
| | First-in detection response to deactivation (Page 187) | Intlk02, Intlk04, Intlk08, Intlk16 |
| 22 | Update acknowledgment and error status of the message call (Page 171) | AssetM, AV, ConPerMon, CountOh, CoutnScL, DoseL, Event, EventTs, FmCont, FmTemp, MonAnL, MonDi08, MonDiL, MotL, MotRevL, MotSpdCL, MotSpdL, OpAnL, PIDConL, PIDConR, PIDStepL, TotalL, Vlv2WayL, VlvAnL, VlvL, VlvMotL |
| 23 | Evaluation of signal status (Page 152) | Intlk02, Intlk04, Intlk08, Intlk16 |
| | Specifying the influence of the signal status on the dosing process (Page 158) | DoseL |
| | SP following PV in open loop has no priority over SP_Ext and SP limits (Page 193) | PIDConL, PIDConR, PIDConS, PIDStepL, FmTemp, FmCont |
| | Limits disabled in faceplate (Page 206) | CntOhSc, CountOh, CountScL, TotalL, MonAnL, MonAnS, MotL, MotRevL, MotSpdCL, MotSpdL, VlvAnL, VlvMotL, VlvPosL |
| 24 | Enabling local operator authorization (Page 169) | ConPerMon, CountOh, CountScL, DoseL, FmCont, FmTemp, GainSched, Intlk02, Intlk04, Intlk08, Intlk16, ModPreCon, MonAnL, MonAnS, MonDi08, MonDiL, MonDiS, MotL, MotS, MotRevL, MotSpdCL, MotSpdL, OpAnL, OpDi01, OpDi03, OpTrig, PIDConL, PIDConR, PIDStepL, Ratio, SelA16In, TotalL, Vlv2WayL, VlvAnL, VlvL, VlvS, VlvMotL, KalFilt |
| | With acknowledge overdosage (Page 190) | DoseL |
| 25 | Suppression of all messages (Page 185) | ConPerMon, DoseL, FmCont, FmTemp, MonAnL, MonAnS, MonDiL, MonDiS, MotL, MotS, MotRevL, MotSpdCL, MotSpdL, PIDConL, PIDConR, PIDStepL, Vlv2WayL, VlvAnL, VlvL, VlvS, VlvMotL |

| Bit number | Meaning | Block |
|---|---|---|
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) | ConPerMon, FmCont, FmTemp, PIDConL, PIDConR, PIDStepL, AV, MotL, MotRevL, MotSpdL, VlvMotL, MotSpdCL, VlvAnL, DoseL, CountOh, CountScL, TotalL, MonAnL, MonAnS, CntOhSc, PIDConS |
| 27 | Selecting values associated with messages (Page 167) | Event, EventNck, EventTs |
| | Interlock display with LocalSetting 2 or 4 (Page 189) | MotL, MotS, MotRevL, MotSpdCL, MotSpdL, Vlv2WayL, VlvAnL, VlvL, VlvS, VlvMotL |
| | Enable configuration of the dribbling quantity (Page 170) | DoseL |
| | Do not evaluate the first run of the diagnostic blocks (Page 205) | FbAnIn, FbAnOu, FbAnTot, FbDiIn, FbDiOu, FbDrive, FbEnMe, FbSwtMMS, Pcs7AnIn, Pcs7AnOu, Pcs7Cnt1, Pcs7Cnt2, Pcs7Cnt3, Pcs7DiIn, Pcs7DiIT, Pcs7DiOu, Pcs7AITC, Pcs7DiCnt |
| 28 | Output invalid raw value (Page 185) | Pcs7AnIn, Pcs7DiIn, Pcs7DiIT, Pcs7AITC, Pcs7DiCnt |
| | Disabling operating points (Page 156) | AssetM, AV, ConPerMon, CountOh, CountScL, DoseL, FmCont, MonAnL, PIDConL, PIDStepL, VlvAnL, FmTemp, PIDConR, MotL, MotRevL, MotSpdCL, MotSpdL, TotalL, VlvMotL |
| | Reading messages (Page 158) | FbDrive |
| 29 | Output substitute value if raw value is invalid (Page 160) | FbAnIn, FbDiIn, Pcs7AnIn, Pcs7DiIn, Pcs7DiIT, Pcs7AITC, Pcs7DiCnt |
| | Signaling limit violation (Page 181) | AV, ConPerMon, CountOh, CountScL, DoseL, FmCont, FmTemp, MonAnL, MotL, MotRevL, MotSpdCL, MotSpdL, PIDConL, PIDConR, PIDStepL, TotalL, VlvAnL, VlvMotL |
| | Transmission of status information of devices (Page 186) | FbDrive, FbSwtMMS |
| | Output substitute values in case of communications error (Page 204) | Rcv_AnaVal, Rcv_DigVal |

| Bit number | Meaning | Block |
|---|---|---|
| 30 | Outputting last valid value if raw value is invalid (Page 165) | FbAnIn, FbDiIn, Pcs7AnIn, Pcs7DiIn, Pcs7DiIT, Pcs7AITC, Pcs7DiCnt |
| | Outputting a de-energized value for block-external simulation (Page 159) | FbAnOu, FbDiOu, Pcs7AnOu, Pcs7DiOu, FbDrive, FbSwtMMS |
| | Set reset depending on the operating mode or the LiOp parameter (Page 174) | DoseL, MotL, MotS, MotRevL, MotSpdL, MotspdCL, VlvL, VlvS, VlvMotL, VlvAnL, Vlv2WayL, CountScL, CountOh, CntOhSc; TotalL |
| | Applying the dynamically activated dead band during the PV settling time (Page 152) | PIDConL, PIDConR |
| | Output last values in case of communications error (Page 204) | Rcv_AnaVal, Rcv_DigVal |
| 31 | Activating recording of the first signal (Page 161) | Intlk02, Intlk04, Intlk08, Intlk16 |
| | Activating reset of protection / error in manual mode (Page 175) | DoseL, MotL, MotS, MotRevL, MotSpdL, MotspdCL, VlvL, VlvS, VlvMotL, VlvAnL, Vlv2WayL |
| | Limits disabled in faceplate (Page 206) | FmCont, FmTemp, PIDConL, PIDConR, PIDConS, PIDStepL, ConPerMon |
| | Keep last valid raw value if input value is bad (Page 207) | Pcs7AnOu, Pcs7DiOu, FbAnOu, FbDiOu |

### See also

Flow setpoints in percent (Page 157)

Use an internal or external setpoint for the absolute fine dosing quantity (Page 164)

Polygon functions (Page 2290)

## 2.1.9.2 Stopping dosing at a flow alarm

### Feature bit

Number of the `Feature` bit: 11

### Stopping dosing at a flow alarm

You can use this `Feature` bit to enable stopping dosing at a flow alarm.

The default setting is 0.

**Bit = 0**: Disabled, dosing is not stopped when a flow alarm occurs

**Bit = 1**: Enabled, dosing is stopped when a flow alarm occurs

### 2.1.9.3 Failure handling

#### Feature bit

Number of the `Feature` Bit: 6

#### Failure handling

Use this `Feature` bit to configure "Failure handling" in the case of device failure.

The default setting is 0.

**Bit = 0**: Commands `Local` and `Ackn` only active

**Bit = 1:** All commands active

#### See also

Configurable functions using the Feature I/O (Page 139)

### 2.1.9.4 Set startup characteristics

#### Feature bit

Number of the `Feature` bit: 0

#### Set startup characteristics

With this `Feature` bit, you set the startup characteristics of the function blocks, for example, for:

- Motors, valves and controllers
- Channel blocks
- Monitoring blocks, e.g. `MonAnL` and `MonDiL`.
- Mathematical and analog logic blocks
- The OpAnL, OpAnS, OpDi01, and OpDi03 block
- The TimeTrg block
- Dosing block DoseL
- Counter blocks
- The Average block
- The TimeTrig block
- The Trigger block

The default setting is 0.

**Note**

This `Feature` bit has no function in the "Out of service" operating mode. The process tag remains in the "Out of service" operating mode after a warm restart of the CPU.

**Note**

The message classes Alarm, Warning and Tolerance are not valid for user-configured message classes. Please take into consideration the validity of terms for user-configured message classes (Page 45).

**Note**

With a Run-Stop-Run transition of the CPU and internally pending messages, non-stuck-through messages with time stamps and auxiliary values beginning with `RunUpCycle` occur for blocks with the startup characteristic `Feature` bit = 0 after expiration of the `RunUpCycle` counter in the following cases:

- Alarm, warning or tolerance messages from the operating points (motor, valve, dosing, controller and analog monitoring blocks)
- Feedback errors (motor and valve blocks)
- Output signals of digital process tags (MonDiL, MonDi08)
- Flutter limits violated (MonDiL)

The restart routines of the blocks reset the following outputs in OB100:

- Operating point outputs `xx_AH_Act`, `xx_AL_Act`, `xx_WH_Act`, `xx_WL_Act`, `xx_TH_Act`, `xx_TL_Act` or `GradHUpAct`, `GradHDnAct`, `GradLAct`
- Feedback error outputs `MonDynErr` and `MonStaErr`
- Output binary signals `Out`, `Out1..8` for MonDiL or MonDi08
- Flutter suppression `FlutAct` for MonDiL

This causes an outgoing message when initializing Alarm8_P in OB100 and an incoming message after expiration of the `RunUpCycle` counter on the cyclic interrupt level.

**Note**

During a complete download with AS stop, the blocks (with `Feature.Bit0` = 1) cannot resume operation in their previous mode and control when restarted.

### Set startup characteristics for motors, valves and controllers

**Bit = 0**: Starting the block in manual mode and in neutral position. With controllers, the setpoint is set to (`SP_Int`) internally. See also the section Neutral position for motors, valves and controllers (Page 54) for more on this.

The controller blocks PIDConL, PIDConS, PIDConR, PIDStepL, ModPreCon, MPC10x10, and the valve block VlvAnL have the additional `Feature.Bit16`, Neutral position manipulated variable takes effect at startup (Page 176), which specifies whether the neutral position is approached

**Bit = 1**: Starting the block with the last stored values, in other words in the last operating mode set (manual, automatic or local mode) and at the last valid position.

**Note**

**Special note following complete download to the CPU**

Following a complete download to the CPU, the motor protection signal `Trip` is evaluated during the initial run as good (=1).

When a motor protection signal is pending, this causes a non-struck-through message with time stamp and auxiliary values beginning with `RunUpCycle` after the complete download and after expiration of the `RunUpCycle` counter.

### Startup characteristics for the ShrdResS block

**Bit = 0**: The output command interface is reset to 0.

**Bit = 1**: The block leaves the output command interface unchanged.

### Defining the startup characteristics for channel blocks

**Bit = 0**: The channel block uses either the process value `PV_In` or the value of `SimPV_In` as the startup value, depending on the setting of the input parameter `SimOn` (`PV_In = PV_Out` or `SimPV_In = PV_Out`).

**Bit = 1**: The channel block uses the value `StartVal` as the startup value (`StartVal = PV_Out`).

### Defining the startup characteristics for monitoring blocks

**Bit = 0**: The most recently stored values are reset on startup.

**Bit = 1**: The most recently used value at the output parameter `Out` is output on startup.

### Defining the startup characteristics for mathematical, analog logic blocks

**Bit = 0**: The `Out` output parameter is reset to 0 on startup.

**Bit = 1**: The most recently saved value is output at the `Out` output parameter on startup.

### Defining startup characteristics for the OpAnL/OpAnS blocks

**Bit = 0**: The internal setpoint is used for startup.

**Bit = 1**: The most recently saved value is output at the `Out` output parameter on startup.

### Defining startup characteristics for the OpDi01/OpDi03 blocks

**Bit = 0**: The output parameters `Out` or `Out1`...`Out3` will be reset on startup.

**Bit = 1**: The most recently used value at the output parameter `Out` or `Out1`...`Out3` is outputted on startup.

### Defining the startup characteristics for dosing block DoseL

Bit = 0: On startup, the dosing is ended and reset to the value specified in the input parameter. For example, the dribbling quantity is reset to input `DribbIn`.

Bit = 1: On startup (no complete download), dosing continues with the most recently stored value.
For example, the learned dribbling quantity in output `DribbOut` can further used.

Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter `RunUpCyc`.

**Note**

**Advanced configuration of the startup characteristics for the dosing block**

Note that you can further affect the startup characteristics of the dribbling quantity via the `Feature2.Bit7` as a function of this `Feature.Bit0`. Refer to the sectionUse the last value following a complete download as the current value during startup of the block (Page 165)

### Defining the startup characteristics for counter blocks

**Bit = 0**: On startup, the counter is stopped and reset to the value specified in the input parameter.

**Bit = 1**: On startup, counting continues with the most recently stored value.

Input parameters for the startup characteristics of the counter blocks:

- Block CountOh: Input parameter `PresetTime`
- Block CountScL: Input parameter `PresetVal`
- Block TotalL: Input parameter `PresetVal`

Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter `RunUpCyc`.

**Note**

**Advanced configuration of the startup characteristics for the counter blocks**

Note that you can further affect the startup characteristics via the `Feature.Bit5` as a function of this `Feature.Bit0`. Refer to the section Use the last value following a complete download as the current value during startup of the block (Page 165).

### Defining startup characteristics for the Average block

**Bit = 0**: At startup, averaging begins with the value that is currently at the input parameter `(Out = In, NumCycles = 1)`.

**Bit = 1**: When starting-up, the last `Out` and `NumCycles` values saved are used as the last value for averaging `(Out ≠ In)`.

### Defining startup characteristics for the TimeTrig block

**Bit 0 =0**: Periodic trigger and single trigger can be shut off. The trigger pulse `Trigger` is reset.

**Bit 0 =1**: The activation of the periodic trigger and single trigger as well as the single trigger point are retained.

The InPerTrigOn, InSglTrigOn and InSglTrigDT inputs are applied for this. If you want to use the reset for a complete download, you must read back the marked parameters in addition to the operated and monitored parameters before a complete download.

### See also

User-configured message classes (Page 45)

## 2.1.9.5 Applying the dynamically activated dead band during the PV settling time

### `Feature` bit

Number of the `Feature` bit: 30.

### Dynamic deactivation of the dead band by the `PV` settling time

You can use this `Feature` bit to improve the `PV` settling time in the range of the dead band.

The deadband is deactivated until the `PV` settles more in the center than at the edges of the dead band. The probability that the controlled variable is within the dead zone time is increased, and further control actions are no longer necessary. This leads to reduced wear and less energy consumption.

**Bit = 0**: Dead band width is constant

**Bit = 1:** The dead band is deactivated for the `PV` settling time. When the `PV` settling time is over, the dead zone gets the value of the `DeadBand` parameter.

**Note**

If `Feature.Bit30` = 1 is set, the ConPerMon block has to be connected with the controller. The blocks PIDConL or PIDConR monitor this by checking the status signal of `MeanMV.ST`. In case of `Feature.Bit30` = 1 and `MeanMV.ST` = 16#78 the dynamic deactivation of deadband is not possible and the output `ErrorNum` will be set to 52.

## 2.1.9.6 Evaluation of signal status

### Feature bit

Number of the `Feature` bit: 23

### Evaluation of signal status

You can use the Feature bit to specify if the signal status of the inputs is to be checked for the values 16#00 or 16#28. The signal status of the inputs itself remains unchanged here.

The default setting is 0.

**Bit = 0**: No evaluation of the signal status for 16#00 or 16#28.

**Bit = 1**: the signal status is determined, an input with ST = 16#00 or 16#28 is forwarded with value = 0. The "Negate signal" function at the input of the block has no influence on the reaction in this case.

## 2.1.9.7 Evaluation of the signal status of the interlock signals

Feature / Feature2 bit

Number of the Feature / Feature2 bit: 5

### Family

Family: Operate number of the Feature bit: 5

Family: Control, drives, dosage number of the Feature2 bit: 5

### Evaluation of the signal status of the interlock signals

You can use the Feature / Feature2 bit to specify if the signal status of the interlock inputs is to be checked for the values 16#00 or 16#28. The signal status of the inputs itself remains unchanged here.

The default setting is 0.

**Bit = 0**: the signal status is determined, an input with ST = 16#00 or 16#28 is forwarded with value = 0.

**Bit = 1**: No evaluation of the signal status for 16#00 or 16#28.

## 2.1.9.8 Automatic post dosing for underdosing in automatic mode

### Feature bit

Number of the Feature bit: 12

### Automatic post dosing for underdosing in automatic mode

Use this Feature bit to enable automatic post dosing for underdosing in automatic mode.

The default setting is 0.

**Bit = 0**: Disabled, no automatic post dosing is started for underdosing in automatic mode.

**Bit = 1**: Enabled, automatic post dosing is started for underdosing in automatic mode.

### 2.1.9.9 Block as summing unit or integrator

#### Feature bit

Number of the `Feature` bit: 6

#### Specifying the summing unit or integrator function mode

You can define the summing response of the block via this `Feature` bit.

The default setting is 0.

**Bit = 0**: The block operates as a summing unit.

**Bit = 1**: The block operates as an integrator.

---

**Note**

**Special note for the summing unit**

Note that the characteristics of the block as a summing unit (`Feature.Bit6` = 0) can be further affected via the `Feature.Bit7`.

Refer to the section: Summing characteristics continuous or triggered (Page 184).

---

### 2.1.9.10 Switching operator controls for external setpoint to visible

#### Feature bit

Number of the `Feature` bit: 21

#### Switching operator controls for external setpoint to visible

Use this `Feature` bit to switch all operator controls for the external setpoint to visible in the faceplates for the OS operator. This is always required when the external setpoint should actually be used, for example, for slave controllers in cascade and ration controlling.

The default setting is 0, which keeps the faceplate as clear as possible for simple applications.

**Bit = 0**: The operator controls for the external setpoint are **not visible** in the faceplate.

**Bit = 1**: The operator controls for the external setpoint are **visible** in the faceplate.

### 2.1.9.11 Limit output Out

#### Feature bit

Number of the `Feature` Bit: 5

#### Limit output `Out`

Use this `Feature` bit to define the limit for the output `Out`.

The default setting is 0.

**Bit = 0**: Beyond the end interpolation points, the `Out` output is extrapolated based on the first two or last two interpolation points.

**Bit = 1**: The `Out` output is limited to the first and last interpolation point.

#### See also

Configurable functions using the Feature I/O (Page 139)

### 2.1.9.12 Activating calculation of the flow rate for dosing by scale

#### Feature bit

Number of the `Feature` bit: 7

#### Calculation of the flow rate for dosing by scale

Use this `Feature` bit to activate calculation of the flow rate for dosing by scale.

The default setting is 0.

**Bit = 0**: Deactivated

**Bit = 1**: Calculation activated.

The flow is determined by the change to the dosing quantity per second.

### 2.1.9.13 Condition monitoring information at MOD_Blocks

#### Feature bit

Number of the `Feature` bit: 10

**Bit = 0**: Not forwarded

**Bit = 1**: forwarded

### 2.1.9.14 Condition monitoring information at the channel blocks

#### Feature bit

Number of the `Feature` bit: 10

#### Transferring condition monitoring information to the Maintenance Station

With this `Feature` bit, the data which is entered at the parameters `MS_Ext` and `TextRef` will be transferred to the corresponding driver block and the same data will be displayed on the asset faceplate of the Maintenance Station. For more information, refer to the *Maintenance Handbook*.

The default setting is 0.

**Bit = 0**: Data will not be transferred.

**Bit = 1**: Data will be transferred.

### 2.1.9.15 Disabling operating points

#### Feature bit

Number of the `Feature` bit: 28

#### Disabling operating points

You can use this `Feature` bit to determine if the operating point function of a limit for disabling the message (`MsgLock = 1`) should also be disabled.

The default setting is 0.

**Bit = 0**: Operating point is not suppressed

**Bit = 1**: Operating point is suppressed

### 2.1.9.16 Enabling direct changeover between forward and reverse

#### Feature bit

Number of the `Feature` bit: 7

#### Direct changeover between forward and reverse

With this `Feature` bit, you can enable direct reversal of the direction of motors.

The default setting is 0.

**Bit = 0**: Direct reversal of the direction is disabled.

You can only change the direction of the motor by first stopping and starting the motor again in the required direction. The motor can only be started again after the time set in the `IdleTime` parameter has elapsed.

**Bit = 1**: Direct changeover is enabled.

You can reverse the motor direction directly. The motor block reverses the direction automatically. The motor is stopped and is started in the other direction when the time set in the `IdleTime` parameter has elapsed.

**Note**

In case of automatic / local mode and Bit = 0:

If additional `Feature2.Bit3` =1 Control priority in the event of invalid or not plausible input commands is active, an automatic / local command for reverse direction is still possible. The motor stops for the time of IdleTime (minimum one cycle) then the reverse automatic / local command is plausible and a change of direction take place automatically.

### 2.1.9.17 Specifying the dosing type

#### Feature bit

Number of the `Feature` bit: 5

#### Specifying the dosing type

You can specify the dosing type to be used for the block using this `Feature` bit.

The default setting is 0.

**Bit = 0**: Flow

**Bit = 1**: Scales

### 2.1.9.18 Flow setpoints in percent

#### `Feature` bit

Number of the `Feature` bit: 15

#### Flow setpoints in percent

You can use this `Feature` bit to specify if the doser block should display the flow setpoints in the faceplate as percentages.

The default setting is 0.

**Bit = 0:** Deactivated. The display of the flow setpoints is made in the unit specified with the `PV_Unit` parameter.

**Bit = 1:** Activated: The display of the flow setpoints is made in the unit %.

### 2.1.9.19 Specifying the influence of the signal status on the dosing process

#### Feature bit

Number of the `Feature` bit: 23

#### Specifying the influence of the signal status on the dosing process

You can use this `Feature` bit to specify how the doser should react dependent on the signal status.

The default setting is 0.

**Bit = 0:** Dosing process does not stop with bad signal status of the process value `PV`.

**Bit = 1:** Dosing process stops with bad signal status of the process values `PV`. The doser is also set to the "Off" state (see state diagram: DoseL functions (Page 1312)).

See also the following section:

- Forming and outputting signal status for blocks (Page 116)

### 2.1.9.20 Unit for the rate of change

#### Feature bit

Number of the `Feature` bit: 8

#### Specifying the unit for the rate of change

You can use this `Feature` bit to specify the unit for the rate of change:

The default setting is 0.

**Bit = 0:** unit for the rate of change in the unit of measurement to or from the field device

**Bit = 1:** unit for the rate of change as a percentage to or from the field device

### 2.1.9.21 Reading messages

#### Feature bit

Number of the `Feature` bit: 28

#### Use of the input data

You can use this `Feature` bit to specify the format for reading messages.

Default setting is 0

Bit = 0: Messages are read as PV freely configurable

Bit = 1: Messages are read as MsgNamur

**Note**

This `Feature` bit is only used when "Telegram type 20" and the "PZD 6" parameter are active.

### 2.1.9.22 Setting the scaling for the process values

#### Feature Bit

Number of the Feature Bit: 5

#### Calculation for the outputs CurrentLi, Power1Li, Power2Li, FreeLi

`Feature.Bit5` defines whether the PZDIN**X**Scale is used for the calculation for the other process values.

Bit = 0: **xxxx**.Value := (PZD**X**In*(100/16384.0));

Bit = 1: **xxxx.**Value := (PZD**X**In*(( PZDIn**X**Scale.HIGH- PZDIn**X**Scale.LOW)/16384.0))+ PZDInXScale.LOW;

### 2.1.9.23 Outputting a de-energized value for block-external simulation

#### Feature bit

Number of the `Feature` bit: 30

#### Outputting a de-energized value for block-external simulation

Use this `Feature` bit for channel blocks to specify which value is to be output during the block-external simulation (Simulating signals (Page 64)).

The default setting is 1.

**Bit = 0:** If process value input has the status `16#60`, the output value will be set to the input value.

**Bit = 1:** If process value input has the status `16#60`, the output value will be set to “0”.

### 2.1.9.24 Switch to substitute value

#### Feature bit

Number of the `Feature` bit: 8

### Switch to substitute value

You can use this `Feature` bit to specify whether a substitute value should be applied or the last valid value should be retained when certain conditions occur.

The default setting is 0.

**Bit = 0**: Retain last valid value

**Bit = 1**: Apply the substitute value

### See also

Output invalid raw value (Page 185)

Outputting last valid value if raw value is invalid (Page 165)

## 2.1.9.25 Substitute value switch in the event of an error

### Feature bit

Number of the `Feature` bit: 6

### Substitute value switch in the event of an error

You can specify the substitute value activation at errors using this `Feature` bit.

The default setting is 0.

**Bit = 0**: No substitute value switch

**Bit = 1**: The substitute value `SubV_In` is output at `Out` for `ErrorNum` ≠

### See also

Configurable functions using the Feature I/O (Page 139)

## 2.1.9.26 Output substitute value if raw value is invalid

### Feature bit

Number of the `Feature` bit: 29

### Output substitute value if raw value is invalid

Use this `Feature` bit to activate output of the substitute value (input parameter `SubsPV_In`) for channel blocks if there is an invalid raw value or the device is in the initialization phase (input parameter `PV_ST` 16#4C..4F).

The default setting is 0.

**Bit = 0**: If there is an invalid raw value or the device is in the initialization phase, the substitute value is not output.

**Bit = 1**: If there is an invalid raw value or the device is in the initialization phase, the substitute value is output. The signal status of the output value is set to "Manipulated value (for example, substitute value, simulation, last valid value).

If there is an invalid raw value or the device is in the initialization phase, the output parameter `Bad = 1` is set automatically.

#### Prioritizing the `Feature` bits for channel blocks:

You need to assign parameters for three `Feature` bits for the response to an invalid raw value for the channel blocks.

If more than one of these `Feature` bits are set (=1), the following priority applies:

- Output invalid raw value (Page 185) (`Feature.Bit28` = highest priority)
- Output substitute value if raw value is invalid (`Feature.Bit29`)
- Outputting last valid value if raw value is invalid (Page 165) (`Feature.Bit30` = lowest priority)

The raw value is output of none of the `Feature.Bit28`, `Feature.Bit29` or `Feature.Bit30` is set.

### 2.1.9.27 Activating recording of the first signal

#### Feature Bit

Number of the `Feature` bit: 31

#### Activating recording of the first signal

Use this `Feature` bit to activate recording of the first signal with interlock blocks. Please also refer to the section Recording the first signal for interlock blocks (Page 58).

Default setting is 0

**Bit = 0**: Recording of the first signal is deactivated.

**Bit = 1**: Recording of the first signal is activated.

### 2.1.9.28 External control deviation

#### Feature bit

Number of the `Feature` bit: 14

#### External control deviation

You can use this `Feature` bit to specify whether the external control deviation is to be activated.

The default setting is 0.

**Bit = 0**: The external control deviation is deactivated, the internal control deviation is active.

**Bit = 1**: The external control deviation is activated.

### 2.1.9.29 Activating error state for external process control error CSF

#### Feature bit

Number of the `Feature` bit: 18

#### Activating error state for external process control error CSF

You can use this `Feature` bit to specify whether the block should switch to the error state at an external process control error CSF =1.

The default setting is 0.

**Bit = 0**: : Block does not go to error state with process control error CSF = 1.

**Bit = 1**: Activated: Block goes to error state with process control error CSF = 1.

### 2.1.9.30 Frequency converter with separate device feed

#### Feature bit

Number of the `Feature` bit: 15

#### Frequency converter with separate device feed

You can use this `Feature` bit to specify if a frequency converter with a separate device feed or the inverter enable should be used.

The default setting is 0.

**Bit = 0**: Motor without device feed or without inverter enable

**Bit = 1**: Frequency converter with separate device feed or with inverter enable

### 2.1.9.31 Separate evaluation for excluded and simulated interlock signals

#### Feature bit

Number of the `Feature2` bit: 2

### Separate evaluation for excluded and simulated interlock signals

You can use this `Feature` bit to set the reaction of the block to an interlock signal of an interlock block that is excluded or simulated.

The default setting is 0

**Bit = 0**: Switch-relevant excluded and simulated interlock signals are processed with the status 16#60 and are displayed as simulated or excluded, depending on the interlock status.

**Bit = 1**: Excluded and simulated interlock signals are evaluated separately. All excluded interlock signals in a sequence of interlock blocks are displayed as excluded. Switch-relevant simulated interlock signals are processed with the status 16#60 and displayed as simulated.

**Note**

If the `Feature` Bit is enabled, the signal cannot be inverted via CFC during interconnections at the inputs, as, otherwise, the bypass display would not update.

The interlocking inputs of the following blocks are affected:

- **Intlk02**, **Intlk04**, **Intlk08**, **Intlk16**: `In01, In02, …`
- **DoseL**, **Vlv2WayL**: `Permit, Interlock, Protect`
- **MotL**, **VlvL:** Permit, `Interlock, Protect,PermitSafePos`
- **MotRevL, MotSpdCL**: `Permit, Interlock, Protect , PermRev, IntlRev, ProtRev, PermitSafePos`
- **MotSpdL**: `Permit, Interlock, Protect , PermSp2, IntlSp2, ProtSp2, PermitSafePos`
- **VlvAnL,VlvMotL**, **VlvPosL**: `Permit, Interlock, Protect , PermCls, IntlCls, ProtCls, PermitSafePos`
- **MotS**, **VlvS**, **OpDi01**, **OpDi03**, **PIDConL**, **PIDConR**: `Interlock`

The inversion of the interlocking signals occurs via the `InvIn01`, `InvIn02`, ... inputs on the interlocking blocks.

### See also

OpDi01 functions (Page 428)

OpDi03 functions (Page 443)

Intlk02 functions (Page 1952)

Intlk04 functions (Page 1965)

Intlk16 functions (Page 1997)

Intlk08 functions (Page 1980)

MotL functions (Page 1382)

MotS functions (Page 1419)

MotRevL functions (Page 1447)

MotSpdL functions (Page 1563)

Vlv2WayL functions (Page 1653)

VlvL functions (Page 1696)

VlvMotL functions (Page 1762)

VlvAnL functions (Page 1816)

MotSpdCL functions (Page 1496)

VlvS functions (Page 1732)

DoseL functions (Page 1312)

Influence of the signal status on the interlock (Page 112)

Forming the group status for interlock information (Page 113)

Forming and outputting the signal status for interlock blocks (Page 124)

### 2.1.9.32 Use an internal or external setpoint for the absolute fine dosing quantity

#### Feature Bit

Number of the `Feature` bit: 8

#### Use an internal and external setpoint for the absolute fine dosing quantity

Use this `Feature` bit to determine whether the doser processes the internal and external setpoint for the fine dosing quantity in an absolute manner and is displayed or operated in an absolute manner in the faceplate.

The default setting is 0.

**Bit = 0**: Deactivated. The internal and external I/Os as well as the display and operation of the fine dosing quantity setpoint are processed in the % unit.

**Bit = 1**: Activated: The internal and external I/Os as well as the display and operation of the fine dosing quantity setpoint are processed in the unit that was set using the parameter `DQ_Unit`.

### 2.1.9.33 Activating the run time of feedback signals

#### Feature Bit

Number of the `Feature` bit: 11

#### Activating the run time of feedback signals

Use this `Feature` bit to activate the run time of feedback signals.

The default setting is 0.

**Bit = 0**: Deactivated: Tracking of feedback for simulation immediately after the trigger signal.

**Bit = 1**: Activated: Tracking of feedback for simulation after the trigger signal and expiration of the monitoring time (`MonTiDynamic`). The feedback signals are generated after expiration of the monitoring time.

### 2.1.9.34 Outputting last valid value if raw value is invalid

#### Feature bit

Number of the `Feature` bit: 30

#### Outputting last valid value if raw value is invalid

Use this `Feature` bit to activate output of the last valid value for channel blocks if there is an invalid raw value or the device is in the initialization phase (input parameter PV_ST = 16#4C..4F).

The default setting is 0.

**Bit = 0**: If there is an invalid raw value or the device is in the initialization phase, the last valid value is not output.

**Bit = 1**: If there is an invalid raw value or the device is in the initialization phase, the last valid value is output. The signal status of the output value is set to local "Manipulated value (for example, substitute value, simulation, last valid value).

If there is an invalid raw value or the device is in the initialization phase, the output parameter `Bad = 1` is set.

#### Prioritizing the `Feature` bits for channel blocks:

You need to assign parameters for three `Feature` bits for the response to an invalid raw value for the channel blocks.

If more than one of these `Feature` bits are set (=1), the following priority applies:

- Output invalid raw value (Page 185) (`Feature.Bit28` = highest priority)
- Output substitute value if raw value is invalid (Page 160) (`Feature.Bit29`)
- Output the last valid value if raw value is invalid (`Feature.Bit30`, lowest priority)

The raw value is output of none of the `Feature.Bit28`, `Feature.Bit29` or `Feature.Bit30` is set.

### 2.1.9.35 Use the last value following a complete download as the current value during startup of the block

#### Feature bit

Number of the `Feature` bit: 5

Number of the `Feature2` bit: 7

#### Use the last value during startup of the block after a complete download of the CPU

You can use this `Feature` bit to define the startup characteristics of the block as a function of the `Feature` bit `0`.

This `Feature` bit is used on the following parameters:

- Block TotalL: `OldOut`, `OldCntOut`
- Block CountScL: `OldOut`
- Block CountOh: `OldDays`, `OldHours`, `OldMinutes`, `OldSeconds`
- Block DoseL: `OldDribbOut`

For the settings on `Feature.Bit0`, refer to the section: Set startup characteristics (Page 148).

**Note**

If you want to use this function, you must read back the marked parameters in addition to the operated and monitored parameters before a complete download.

The default setting is 0.

**Bit = 0**: Define the startup characteristics as a function of `Feature.Bit0`:

- Counter blocks:
  - `Feature.Bit0` = 0: The block is set to the default value (input parameter Preset) during startup.
  - `Feature.Bit0` = 1: On startup, counting continues with the most recently stored value.
- Block DoseL:
  - `Feature.Bit0` = 0: The dosing is ended, and the dribbling quantity is set to input `DribbIn`.
  - `Feature.Bit0` = 1: On startup, dosing works with the most recently stored value of `DribbOut`.

**Bit = 1**: During startup of the block after a complete download, the last value `(Oldxxx)` is used as the current value regardless of `Feature.Bit0` ( Set startup characteristics (Page 148)).

### 2.1.9.36 Activate LowCutOff

#### Feature bit

Number of the `Feature` bit: 5

#### Activate LowCutOff

You use this `Feature` bit to activate the use of LowCutOff bits.

The default setting is 0.

**Bit = 0**: Low signal cut-off - OFF (default)

**Bit = 1**: Low signal cut-off - ON

### 2.1.9.37 Selecting values associated with messages

#### Feature Bit

Number of the `Feature` bit: 27

#### Selecting values associated with messages

Use this `Feature` bit to select which values associated with messages are to be output.

The default setting is 0.

**Bit = 0**: The signal status of the binary input is output as the value associated with messages.

**Bit = 1**: The associated analog value is output as the value associated with messages.

### 2.1.9.38 Reporting with BATCH parameters

#### Feature bit

Number of the `Feature` bit: 8

**Reporting with BATCH parameters**

You can use this `Feature` bit to specify whether the block transfers the BATCH parameters

- `BatchID`: Batch ID
- `BatchName`: Batch name
- `StepNo`: Batch step number

Are transferred as associated values to the OS during messaging.

The default setting is 0.

**Bit = 0**: The block does not transfer any BATCH parameters to the OS.

**Bit = 1**: The block transfers BATCH parameters to the OS.

---

**Note**

**Information about the setting "Bit = 1:**

The Event block can no longer transfer the `In8.ST` or `AV8.Value` as an associated value.

The EventTS block can no longer transfer the signal status of the signal `In7` and `In8` or `InTS7` and `InTS8` as an associated value.

---

You will find more information in the following chapters:

- Event messaging (Page 2023)
- EventTs messaging (Page 2049)

### 2.1.9.39 Motor feedback is not available

#### Feature bit

Number of the `Feature` bit: 12

#### Motor feedback is not available

You can use this `Feature` bit to specify whether or not motor feedback should be available.

The default setting is 0.

**Bit = 0**: Motor feedback is available.

**Bit = 1:** Motor feedback is not available. The motor feedback is derived from the control signals internally in the block; this disables the monitoring of the motor feedback.

### 2.1.9.40 Display only input values that are interconnected in the faceplate

#### Feature bit

Number of the `Feature` bit: 5

#### Display only input values that are interconnected in the faceplate

You can use this `Feature` bit to specify whether only interconnected input values `In01` to `In16` (status not equal to 16#FF) should be displayed in the faceplate.

The default setting is 0.

**Bit = 0**: Display all input values.

**Bit = 1**: Display only interconnected input values (status not equal to 16#FF) and in case inputs are not connected, the faceplate size should shrink to the last connected input.

### 2.1.9.41 Activate OS_Perm bits

#### Feature bit

Number of the `Feature` bit: 5

#### Display only input values that are interconnected in the faceplate

You use this `Feature` bit to activate the use of additional `OS_Perm` bits in the faceplate.

The default setting is 0.

**Bit = 0**: `OS_Perm` bits XXX inactive

**Bit = 1**: `OS_Perm` bits XXX active

**With XXX**:

| Block | OS_Perm.BitX |
|---|---|
| Intlk02 | 16-17 |
| Intlk04 | 16-19 |
| Intlk08 | 16-23 |
| Intlk16 | 16-31 |

| OS_Perm.Bit16 | OS_Perm.Bit0 | OS_Perm.Bit1 | Operation exclusion "Set" In01 | Operation exclusion "Reset" In01 |
|---|---|---|---|---|
| 0 | 0 | 0 | No | No |
| 0 | 0 | 1 | No | Yes |
| 0 | 1 | 0 | Yes | No |
| 0 | 1 | 1 | Yes | Yes |
| 1 | X | X | Yes | Yes |

### 2.1.9.42 Disabling opening and closing

#### Feature bit

Number of the `Feature` bit: 6

#### Disabling the Open and Close commands

Only applicable when no auxiliary valve is used (bit 5 = 0)

**Bit = 0:** Open and Close commands affect control valve

**Bit = 1:** Disable the Open and Close commands

### 2.1.9.43 Enabling local operator authorization

#### Feature bit

Number of the `Feature` bit: 24

#### Enabling local operator authorization

You can use this `Feature` bit to enable local permission for a technologic block.

The default setting is 0.

**Bit = 0**: Disabled

**Bit = 1**: Enabled

For information on this operator permission, refer to the section Operator control permissions (Page 285).

### 2.1.9.44 Enable configuration of the dribbling quantity

**Feature bit**

Number of the `Feature` bit: 27

#### Enable configuration of the dribbling quantity

With this `Feature` bit, you can enable configuration of the dribbling quantity during dosing.

The default setting is 0.

**Bit = 0**: Disabled, the dribbling quantity can only be configured in the "End" state or with the automatic determination of the dribbling quantity (`DribbCor` = 1).

**Bit = 1**: Enabled, the dribbling quantity can always be configured, but not in the "Dribbling" state. The automatic determination of the dribbling quantity must be switched off (`DribbCor` = 0).

### 2.1.9.45 Enabling program mode

#### Feature bit

Number of the `Feature` bit: 19

#### Enabling program mode

You can use this `Feature` bit to specify whether or not the controller block should be used for program mode.

Default setting is 0.

**Bit = 0**: The block is not intended for program mode.

**Bit = 1**: The block can be used for program mode. The operator control elements required for this are then visible in the faceplate.

**Note**

This `Feature` bit only applies to the PIDConR block.

#### What is the program mode?

Program mode provides primary controller functions (external Advanced Control software package), which run on an external PC as an OPC client, the option of using the control from the controller function block and specifying the setpoint or manipulated variable from a remote location.

### 2.1.9.46 Update acknowledgment and error status of the message call

#### Feature bit

Number of the `Feature` bit: 22

#### Update acknowledgment and error status of the message call

You can use the `Feature` bit to determine if the acknowledgment and error status of the message call at the block output should be updated.

The default setting is 0.

- **Bit = 0**: The `MsgErr`, `MsgStat` and `MsgAckn` block outputs are set to the default settings and not updated. The block will run faster with this setting.
- **Bit = 1**: The `MsgErr`, `MsgStat`and `MsgAckn` block outputs re updated based on the feedback of the lower level message blocks. The lower level message blocks are called every other cycle as long as an acknowledgment is expected or error information is pending.

### 2.1.9.47 Control zone with frozen I component

#### Feature bit

Number of the `Feature` bit: 13

#### Freezing the I component for control with control zone

You can use this `Feature` bit to specify how the controller should react to control zone operation. The I component must be enabled for this TI <> 0.

The default setting is 0.

**Bit = 0**: The I component is set in such a way that the manipulated variable reacts bumplessly when the error signal reoccurs in the control zone.

**Bit = 1**: If the error signal is outside the control zone, the I component is frozen internally. When it enters the control zone, the controller starts with the value of the I component that was in effect when it left the control zone.

The reaction depends on additional `Feature` bits and controller settings. See also the following section:

- Using control zones (Page 220)

### 2.1.9.48 Control zone with specified I component

#### Feature bit

Number of the `Feature` bit: 12

### Specifying I component for control with control zone

You can use this `Feature` bit to specify how the controller should react to control zone operation. The I component must be enabled for this TI <> 0.

The default setting is 0.

**Bit = 0**: The I component is set in such a way that the manipulated variable reacts bumplessly when the error signal reoccurs in the control zone.

**Bit = 1**: If the error signal is outside the control zone, the I component is set internally as follows:

- `I_Part := MV_Offset`.

The reaction depends on additional `Feature` bits and controller settings. See also the following section:

Using control zones (Page 220)

## 2.1.9.49 Resetting the commands for changing the mode

### Feature Bit

Number of the `Feature` bit: 2

### Resetting the commands for changing the mode

Using this `Feature` bit, you define how the block handles the incoming control commands:

- `SP_IntLi` and `SP_ExtLi` for controllers
- `AutModLi` and `ManModLi` for controllers, drives, and dosage block
- `SaPo_Ext` for KalFilt block
- `OffLi`, `UpLi`, and `DnLi` for TotalL block

The default setting is 0.

**Bit = 0**: The control commands are not reset by the block. If there are two pending control commands for changing mode, the mode is not changed. In this case, the note text "Invalid command" is displayed in the faceplate.

**Bit = 1**: The control commands are reset by the block. This, for example, ensures that if a control command is sent from the SFC, the command is reset automatically after a step is exited.

## 2.1.9.50 Enabling resetting of commands for the control settings

### Feature bit

Number of the `Feature` bit: 3

### Enabling resetting of commands for the control settings

With this `Feature` bit, you select how the block handles commands for the control settings (for example motor on) via the interconnected input parameters.

The default setting is 0.

**Bit = 0**: The control commands are not reset by the block. If there are two commands relating to the control settings at the same time, the status of the control settings is retained. In this case, the "Invalid signal" message is displayed in the standard view of the faceplate.

**Bit = 1**: The control commands are reset by the block. This, for example, ensures that if a control command is sent from the SFC, the command is reset automatically after the step is exited.

## 2.1.9.51 Resetting the dosing quantity when dosing starts

### Feature bit

Number of the `Feature` bit: 6

### Resetting the dosing quantity when dosing starts

Use this `Feature` bit to enable resetting the dosing quantity when dosing starts.

The default setting is 0.

**Bit = 0**: Disabled, the dosing quantity is not reset when dosing starts

**Bit = 1**: Enabled, the dosing quantity is reset when dosing starts.

## 2.1.9.52 Process value with separate scale range

### Feature bit

Number of the `Feature` bit: 16

### Process value with separate scale range

You can use this `Feature` bit to assign a separate scale range to the process value. The default setting is 0.

**Bit = 0**: Apply scale range from `SP`.

**Bit = 1**: `PV` has a separate scale range.

### See also

Neutral position for motors, valves and controllers (Page 54)

Neutral position manipulated variable takes effect at startup (Page 176)

Use the last value following a complete download as the current value during startup of the block (Page 165)

### 2.1.9.53 Resetting via input signals in the event of interlocking (Protection) or errors

#### Feature Bit

Number of the `Feature` bit: 9

#### Resetting with automatic commands (e.g. `OpenAut`, `CloseAut`)

With this `Feature` bit, you define that an automatic control with its input commands (e.g. `OpenAut`, `CloseAut`) also resets an interlock protection or an error.

The default setting is 0.

**Bit = 0**: No resetting.

**Bit = 1**: Reset an interlock protection or an error with a 0-1 edge change in the inputs for automatic commands (e.g. `OpenAut`, `CloseAut`).

### 2.1.9.54 Set reset depending on the operating mode or the LiOp parameter

#### `Feature` Bit

Number of the `Feature` Bit: 30

#### Set reset depending on the operating mode (motor, valve and dosing blocks)

When the "Protection" interlock, feedback error ("Status error", "Control error") or "Motor protection" signal is present again, use this `Feature` bit to specify if a reset can be made depending on the mode only by the operator in manual mode or only by the automatic I/Os in automatic mode.

Resetting to manual mode is enabled with `Feature.Bit31` (Activating reset of protection / error in manual mode (Page 175)). Also refer to the Resetting the block in case of interlocks or errors (Page 47) chapter.

The default setting is 0.

**Bit = 0:** Reset does not depend on the operating mode

**Bit = 1:** In manual mode, manual reset by the operator is only possible if `Feature.Bit31` is set, otherwise no reset is required in manual mode.

In automatic mode, reset can only be made with automatic I/Os, regardless of `Feature.Bit31`. This is performed either with a 0-1 edge transition at the `RstLi` input or, when `Feature.Bit9` is set, with a 0-1 edge transition at the automatic inputs, for example `OpenAut`, `CloseAut`.

**Note**

Rapid stop is unlocked for all operating modes using the "Reset" button in the faceplate (`RstOp = 1`); in CFC it is unlocked using the `RstLi = 1` input parameter.

**Note**

The local operating mode does not depend on this Feature Bit and has a separate reset mechanism.

### Resetting the dosing mode depending on the operating mode (dosing blocks)

You can also use this Feature bit to set whether or not the reset of the dosing quantity is dependent on the operating mode.

The default setting is 0.

**Bit = 0:** Reset does not depend on the operating mode

**Bit = 1:**In manual mode, only a manual reset by the operator is possible at the input RstDQ_Op. In automatic mode, a reset is only possible by a 0-1 edge transition at the input RstDQ_Li.

**Note**

The operating mode Local is independent of this Feature bit. The dosing quantity can be reset by the operator at the input RstDQ_Op or via a 0-1 edge transition at the input RstDQ_Li.

### Set reset depending on the LiOp parameter (counter blocks)

You can use this Feature Bit to determine whether the setting or resetting to a default value should be made dependent on the LiOp parameter.

The default setting is 0.

**Bit = 0:** Setting or resetting to a default value does not depend on the LiOp parameter.

**Bit = 1:** Setting or resetting to a default value depends on the LiOp parameter.

LiOp **=0:** Setting or resetting to a default value can only be made via the faceplate or at the parameter for the faceplate.

LiOp **=1**: Setting or resetting to a default value can only be made at the parameter for interconnections.

## 2.1.9.55 Activating reset of protection / error in manual mode

### Feature bit

Number of the Feature bit: 31

### Activating reset of protection / error in manual mode

Use this Feature bit to specify whether a reset is necessary once the "Protection" interlock signal, feedback errors ("Runtime error", "Control deviation"), or "Motor protection" are present again. See also the following section: Resetting the block in case of interlocks or errors (Page 47).

The default setting is 0.

**Bit = 0:** No reset required in manual mode.

**Bit = 1:** Reset required in manual mode. The reset is performed using the "Reset" button (`RstOp = 1`) or, in CFC, using the input parameter `RstLi`.

**Note**

Rapid stop is unlocked for all operating modes using the "Reset" button in the faceplate (`RstOp = 1`); in CFC it is unlocked using the `RstLi = 1` input parameter.

**Note**

The local operating mode has a separate reset mechanism.

### 2.1.9.56 Reset even with locked state

#### Feature Bit

Number of the `Feature` bit: 19

#### Reset even with locked state

With this `Feature` bit, you specify if it is possible to perform a reset with an active "Protection" or "Motor protection" type interlock. This can be used, for example, to reset hardware interlocks.

The default setting is 0.

**Bit = 0**: No reset is possible with a "Protection" type interlock or with active motor protection.

**Bit = 1**: Reset is possible with a "Protection" type interlock or with active motor protection.

### 2.1.9.57 Neutral position manipulated variable takes effect at startup

#### Feature Bit

Number of the `Feature` bit: 16

#### Neutral position manipulated variable takes effect at startup

You can use this `Feature` bit to specify if the block should go to the neutral position at startup.

Default setting is 0

**Bit = 0**: The block does not go to the neutral position at startup

With control valve VlvAnL, the main valve is closed and the auxiliary valve (if configured) opened.

**Bit = 1**: The block goes to the neutral position at startup

With control valve VlvAnL, the main valve and auxiliary valve (if configured) go to the neutral position.

You can find more information in Section Neutral position for motors, valves and controllers (Page 54).

**Note:**

`Feature.Bit16` is only effective when `Feature.Bit0` = 0.

### 2.1.9.58 Neutral position manipulated variable takes effect with "out of service" operating mode

#### Feature Bit

Number of the `Feature` bit: 15

#### Neutral position manipulated variable takes effect with "out of service" operating mode

You can use this `Feature` bit to specify if the block should go to the neutral position when it transitions to the "Out of service" operating mode.

Default position is "0".

**Bit = 0**: The block does not go to the neutral position at the transition to the "out of service" operating mode.

**Bit = 1**: The block goes to the neutral position at the transition to the "out of service" operating mode.

Refer to the Neutral position for motors, valves and controllers (Page 54) section for more information.

### 2.1.9.59 Setting switch or button mode

#### Feature bit

Number of the `Feature` bit: 4

#### Setting switch or button mode (input signal as pulse signal or as static signal)

You can use this `Feature` bit to determine whether a separate interconnectable 1-active control input has to be used for every automatic command of the block or two automatic commands are assigned to a control input.

The `Feature` bit impacts the following control inputs for automatic mode in the form of a pulse (push-button operation) or a static signal (switch operation):

- Starting and stopping a motor
- Opening and closing a valve

- Switching modes: Automatic and Manual (parameters `AutModLi` and `ManModLi`)
- Setpoint specification: Internal and External (parameters `SP_ExtLi` and `SP_IntLi`)

is given in the form of a pulse (pushbutton operation) or a static signal (switching mode).

**Note**

The input commands are checked for plausibility. For more information, see Error handling (Page 128), e.g. MotL error handling (Page 1390) and `Feature2.Bit3` (Control priority in the event of invalid or not plausible input commands (Page 186)).

You can find the commands for controlling the block in the relevant section on block operating modes. They are always the parameters that are used for the automatic operation of a block.

**Bit = 0**: Button mode: Each automatic command is assigned to a control input. This has a latching reaction and is 1-active.

**Example with a motorMotRevL**: In this case, use the interconnectable input parameters.

- `FwdAut` = `1` for the command "Start forward"
- `RevAut` = `1` for the command "Start backwards"
- `StopAut` = `1` for the stop command and
- `AutModLi` = `1` for setting "Automatic" operating mode
- `ManModLi` = `1` for setting "Manual" operating mode

**Bit = 1**: Switching mode: two static automatic commands are assigned to a control input.

**Example with a motorMotRevL**: In this case, use the interconnectable input parameters.

- `FwdAut` = `1` for the command "Start forward"
- `RevAut` = `1` for the command "Start backwards" and
- `FwdAut` = `0` and `RevAut` = `0` for the stop command
- `AutModLi` = `1` for setting "Automatic" operating mode
- `AutModLi` = `0` for setting "Manual" operating mode

The `StopAut` and `ManModLi` control inputs are irrelevant in this case.

### 2.1.9.60 Specifying switching mode

#### Feature Bit

Number of the `Feature` bit: 5

#### Specifying switching mode

Use this `Feature` bit to specify switching mode for the motor block.

The default setting is 0.

**Bit = 0**: Switching mode "On over speed 1" with the `SwOverTi > 0` parameter

**Bit = 1**: Switching mode "Off over speed 1" with the `SwOverTi > 0` parameter

**Note**

This `Feature` bit is only used with the `SwOverTi > 0` parameter. Refer also to the MotSpdL functions (Page 1563) section for more on this.

### 2.1.9.61 Creep rate is always detected in the dosing quantity

#### Feature Bit

Number of the `Feature` bit: 13

#### Enable: Creep rate is always detected in the dosing quantity

Use the `Feature` bit to specify the response for detecting the creep rate in the dosing quantity.

The default setting is 0.

**Bit = 0**: Disabled, the creep rate is only detected in the dosing quantity over the limit `CR_AH_Lim` (high alarm for creep rate). With `CR_AH_En = 0`, the creep rate has no effect on the dosing quantity calculation.

**Bit = 1**: Enabled, the creep rate is always detected in the dosing quantity.

**Note**

Creep rate is the flow in the states "End", "Off", and "Pause".

### 2.1.9.62 Enabling rapid stop via faceplate

#### Feature bit

Number of the `Feature` bit: 14

#### Enabling rapid stop via faceplate

You can use the `Feature` bit "Enable rapid stop via faceplate" to specify if the OS operator can use rapid stop for the block via the standard view of the faceplate.

The default setting is 0.

**Bit = 0**: The "Rapid stop" button is not visible in the faceplate.

**Bit = 1**: The OS operator can use the button for rapid stop.

### 2.1.9.63 Position feedback signals are active

**Feature bit**

Number of the `Feature` bit: 12

### Position feedback signals are active

You can use this `Feature` bit to activate the position feedback signals (`FbkP1` and `FbkP2`) and deactivate the separate valve feedback signals (`FbkV0`, `FbkV1` and `FbkV2`). In the trend view and preview, the position feedback signals and control signals `Pos0Out, Pos1Out, Pos2Out` are used for display.

Default: 0

**Bit = 0**: The valve feedback signals `FbkV0`, `FbkV1`, `FbkV2` are active. The position feedback signals `FbkP1`, `FbkP2` are deactivated.

**Bit = 1**: The position feedback signals `FbkP1`, `FbkP2` are activate. The valve feedback signals `FbkV0`, `FbkV1`, `FbkV2` are deactivated.

**Note**

`SafeV0`, `SafeV1, SafeV2` and `DefPos1`, `DefPos2` are only used for configuring the icon displays in the OS.

## 2.1.9.64 Separate monitoring time for stopping the motor

### Feature bit

Number of the `Feature` bit: 13

### Separate monitoring time for stopping the motor

You can use this `Feature` bit to activate a separate monitoring time for stopping the motor.

The default setting is 0.

**bit = 0:** A monitoring time for starting and stopping the motor

**Bit = 1**: Separate monitoring time for stopping the motor

## 2.1.9.65 Separate interlock for each direction, position, or speed

### Feature bit

Number of the `Feature`2 bit: 16

### Separate interlock for each direction or position

You can use this `Feature` bit to enable the use of the "Separate interlock for each direction, position, or speed" function.

Default setting is 0

**Bit = 0**: One interlock for each direction, position, or speed.

**Bit = 1**: Separate interlock for each direction, position, or speed.

### 2.1.9.66 Separate delay times for each alarm

#### Feature bit

Number of the `Feature` bit: 8

#### Separate delay times for each alarm

You can use this `Feature` bit to activate the alarm delay type Two time values for each individual limit (Page 228).

The default setting is 0.

**Bit = 0**: The alarm delay type Two time values for each individual limit (Page 228) is deactivated.

**Bit = 1**: The alarm delay type Two time values for each individual limit (Page 228) is activated.

#### See also

User-configured message classes (Page 45)

Two time values per limit pair (Page 227)

### 2.1.9.67 Signaling limit violation

#### Feature bit

Number of the `Feature` bit: 29

#### Signaling limit violation

With this `Feature` bit, you specify how limit violation should be sent to the respective limit outputs.

The default setting is 0.

**Bit = 0**: Output value of the limit output = 1 (1 active)

**Bit = 1**: Output value of the limit output = 0 (0 active)

**Note**

To learn about the parameters with which you can influence this behavior, refer to the description of the connections for the respective blocks.

### 2.1.9.68 Alarm setpoint difference

#### Feature bit

Number of the `Feature` bit: 5

#### Setpoint difference should be alarmed

Use this `Feature` bit to activate alarming when a setpoint difference occurs.

The default setting is 0.

**Bit = 0:** Disabled: The message for the `ExtMsg2` and `ExtMsg3` parameters are output.

**Bit = 1:** Enabled: The messages for the `ER_H_Lim` or `ER_L_Lim` parameter are output instead of the message parameters `ExtMsg2` and `ExtMsg3`.

### 2.1.9.69 Setpoint specification with separate display area and custom unit

#### Feature bit

Number of the `Feature` bit: 16

#### Setpoint specification with separate display area and custom unit

You can use this `Feature` bit to specify the display area and the unit of the setpoint specification.

The default setting is 0.

**Bit = 0**: Setpoint specification and readback value have the same display area (`RbkOpScale`, `RbkUnit`)

**Bit = 1**: The setpoint specification has a separate display area and a custom unit (`SP_OpScale`, `SP_Unit`)

### 2.1.9.70 Control via auxiliary valve

#### Feature Bit

Number of the `Feature` bit: 5

#### Control via a auxiliary valve

You can use this `Feature` bit to specify if control should be performed by a auxiliary valve.

The default setting is 0.

**Bit = 0**: Control without auxiliary valve / no auxiliary valve present

**Bit = 1**: Control via auxiliary valve

### 2.1.9.71 Enabling bumpless change to the proportional gain, derivative time and amplification of the differentiator

#### Feature Bit

Number of the `Feature` bit: 20

#### Enabling bumpless change to the proportional gain, derivative time and amplification of the differentiator

Use this `Feature` bit to enable bumpless change to proportional gain `Gain`, derivative action time `TD` and gain of the differentiator in automatic mode.

The default setting is 0.

**Bit = 0**: The bumpless switchover is deactivated.

**Bit = 1**: The bumpless switchover is activated.

### 2.1.9.72 Enable bumpless switchover to "Automatic" mode for operator only

#### Feature bit

Number of the `Feature` bit: 21

#### Enable bumpless switchover to automatic mode for operator only

You can use this `Feature` bit to specify whether bumpless switchover to automatic mode for valves, motors and dosing unit should be enabled only if switching via the faceplate is in effect, or whether switching using interconnectable inputs `AutModLi` and `ManModLi` (`ModLiOp = 1`) is also possible.

The default setting is 0

**Bit = 0**: The function "Bumpless switchover to automatic mode for valves, motors and dosers" works when switching via the faceplate and switching using interconnectable inputs `AutModLi` and `ManModLi` (`ModLiOp = 1`).

**Bit = 1**: Bumping switchover can be activated via the inputs `AutModLi` and `ManModLi` (`ModLiOp = 1`)

### 2.1.9.73 Disabling bumpless switchover to automatic mode for controllers

#### Feature Bit

Number of the `Feature` bit: 18

### Changeover with or without P step change when the internal setpoint is not in tracking mode

Use this `Feature` bit to specify if a changeover should occur with or without a P step change when the internal setpoint (`SP_TrkPv = 0`) does not track the process value.

The default setting is 0.

**Bit = 0**: Changeover without P step change (bumpless)

**Bit = 1**: Changeover with P step change (not bumpless)

For more detailed information, refer to the description of Manual and automatic mode for control blocks (Page 79).

## 2.1.9.74 Enabling bumpless switchover to automatic mode for valves, motors, and dosers

### Feature bit

Number of the `Feature` bit: 17

### Bumpless switchover

You can use this `Feature` bit to enable the bumpless switchover from local/manual mode to automatic mode.

Default setting is 0

**Bit = 0**: Bumpless switchover is disabled. You can switch from local/manual mode to automatic mode at any time.

**Bit = 1**: Bumpless switchover from local/manual mode to automatic mode is enabled. A switchover from local/manual mode to automatic mode is only possible if the control settings of the local/manual mode and automatic modes match. If switchover occurs at a different point in time, this is indicated in the faceplate with the text "Switchover error".

Refer to the Manual and automatic mode for motors, valves and dosers (Page 82) section for more on this.

A second `Feature` bit is used to specify if pure bumpless switchover to automatic mode is only possible via the faceplate or if switching is also possible via the interconnectable parameters `AutModLi` and `ManModLi` (`ModLiOp = 1`).

Refer to the section Enable bumpless switchover to "Automatic" mode for operator only (Page 183).

## 2.1.9.75 Summing characteristics continuous or triggered

### Feature bit

Number of the `Feature` bit: 7

### Define summing characteristics

You can use this `Feature` bit to define the summing characteristics of the block as a function of the `Feature.Bit6` = 0. If you have set the `Feature.Bit6` = 1, this `Feature` bit has no function.

The default setting is 0.

**Bit = 0**: Triggered summing characteristics.

**Bit = 1**: Continuous summing characteristics.

## 2.1.9.76 Suppression of all messages

### Feature Bit

Number of the `Feature` bit: 25

### Suppression of all messages

You can use this `Feature` bit to determine whether all messages of the block are to be suppressed.

**Bit = 0:** Process messages are suppressed.

**Bit = 1:** All messages are suppressed.

## 2.1.9.77 Output invalid raw value

### Feature bit

Number of the `Feature` bit: 28

### Output invalid raw value

Use this `Feature` bit to activate output of the invalid raw value for channel blocks.

Default setting is 1

**Bit = 0**: The invalid raw value is not output. Either the substitute value (`Feature` bit Output substitute value if raw value is invalid (Page 160)) or the last valid value (`Feature` bit Outputting last valid value if raw value is invalid (Page 165)) is output.

**Bit = 1**: The invalid raw value is output. The signal status of the output value is set to "Bad, device related" or "Bad, process related".

If there is no valid raw value the output parameter `Bad` = `1` is set automatically.

### Prioritizing the `Feature` bits for channel blocks:

You need to assign parameters for three `Feature` bits for the response to an invalid raw value for the channel blocks.

If more than one of these `Feature` bits are set (=1), the following priority applies:

- Output invalid raw value (`Feature.Bit28` = highest priority)
- Output substitute value if raw value is invalid (Page 160) (`Feature.Bit29`)
- Outputting last valid value if raw value is invalid (Page 165) (`Feature.Bit30` = lowest priority)

The raw value is output of none of the `Feature.Bit28, Feature.Bit29 or Feature.Bit30` is set.

### 2.1.9.78 Transmission of status information of devices

#### Feature bit

Number of the `Feature` bit: 29

#### Transmission of status information of devices

You can use this `Feature` bit to specify if status information of the device should be transferred to the upstream diagnostics block.

The default setting is 1.

Bit = 0: No transfer of status information.

Bit = 1: Transfer status information of the device to the upstream diagnostics block.

### 2.1.9.79 Control priority in the event of invalid or not plausible input commands

`Feature2` **Bit**

Number of the `Feature`2 bit: 3

#### Control priority in the event of invalid or not plausible input commands

You can use this function to define the control priority in the event of invalid or not plausible input commands.

Default setting: 0

**Bit = 0**: In the event of invalid or not plausible input commands, the control output is retained.

**Bit = 1**: In the event of invalid or not plausible input commands, the control output switches to the "neutral position".

**Note**

To Bit = 1:

A change of direction in automatic / local mode with block `MotRevL`, `MotSpdCL` is possible although the `Feature.Bit7` =0 Enabling direct changeover between forward and reverse is not parameterized. The motor stops for the time of IdleTime (minimum one cycle) then the reverse automatic / local command is plausible and a change of direction take place automatically.

### 2.1.9.80 Sealing the valve

**Feature Bit**

Number of the `Feature` bit: 8

#### Sealing the valve

You can use this `Feature` to enable the valve sealing function. The torque shutoff is then also evaluated when the valve is closed.

The default setting is 0.

**Bit = 0**: "Seal valve" function is disabled.

**Bit = 1**: "Seal valve" function is enabled.

### 2.1.9.81 First-in detection response to deactivation

#### Feature Bit

Number of the `Feature` Bit: 21

#### First-in detection response to deactivation

You can use this `Feature` Bit to define the first-in detection response of the interlock blocks depending on the `FirstInEn = 0` input parameter.

The default setting is 0.

- **Bit = 0**: When first-in detection is deactivated via `FirstInEn = 0`, the output parameter `FirstIn` is reset.
- **Bit = 1**: When first-in detection is deactivated via `FirstInEn =0`, the output parameter `FirstIn` is maintained. When first-in detection is activated for the first time by an edge change of `FirstInEn 0->1`, the output parameter `FirstIn` is reset.

### 2.1.9.82 Reaction of the switching points in the "Out of service" operating mode

**Feature Bit**

Number of the `Feature` bit: 26

#### Reaction of the switching points in the "Out of service" operating mode

You can use this `Feature` bit to specify the reaction of the switching points to the "Out of service" operating mode.

The default setting is 0.

**Bit = 0**: Last state of the switching points before switching to the "Out of service" operating mode is retained.

**Bit = 1**: The state of the switching points is reset to "Good".

**Note**

The `Feature` "Changes signal status of outputs in OOS" has higher priority over the `Feature` "Reaction of the switching points in the "Out of service" operating mode".

### 2.1.9.83 Reaction to the out of service mode

#### Feature bit

Number of the `Feature` bit: 1

#### Reaction to the out of service mode

You can use this `Feature` bit to define the reaction of the technologic block based on the interconnectable input parameter `OosLi = 1`.

The default setting is 0.

- **Bit = 0**: The symbol for the "In progress" status (see below) appears in the block icon and in the faceplate of the assigned technologic block. A 0-1 edge transition at the input parameter `OosLi` has no further influence on the reaction of the technological block; the previous status is retained. No switch to the "Out of service" mode is performed.
- **Bit = 1**: The mode switches to "Out of service" assuming that the block is "On" or "Manual" mode. If this is not the case, the mode does not change. The symbol for the "In progress" (see below) status also appears in the block icon and in the faceplate of the assigned technologic block regardless of the mode change. No message is output to indicate whether or not the mode change took place.

The status display for "In progress" appears as follows:



A 1-0 edge transition at the input parameter `OosLi` has no influence on the reaction of the technologic block, the previous status is retained.

See also the section Release for maintenance (Page 70) for more on this.

### 2.1.9.84 Exiting local mode

#### Feature bit

Number of the `Feature` bit: 10

#### Reaction to exiting local mode

Use this `Feature` bit to define how "Local mode" is be exited with `LocalSetting` = `1` or `LocalSetting` = `2` and if the mode is not specified by `AutModLi` or `ManModLi`.

Default setting is 0

**Bit = 0**: Exiting local mode in manual mode (bumpless because the control signals are continuously adjusted).

**Bit = 1**: When local mode is exited, the mode changes back to the last mode that was active prior to local mode (not bumpless).

For more detailed information, refer to the description of Local mode (Page 86).

**Note**

**Functionality of "`Feature.Bit10`" is supressed over "`Feature.Bit17`"**

When `Feature.Bit17`=1 and `Feature.Bit10`=1 , after a change from "Automatic Mode" to "Local Mode" the block leaves the "Local Mode" in case of not matching commands to "Manual Mode" and not to "Automatic mode".

### 2.1.9.85 Interlock display with LocalSetting 2 or 4

#### Feature bit

Number of the `Feature` bit: 27

#### Interlock display with LocalSetting 2 or 4

Use this `Feature` bit to specify the display of interlocks with `LocalSetting` `2` or `4` at the faceplate and at the block output `LockAct` .

The default setting is 0

**Bit = 0**: `LocalSetting` `2` and `4` crossed out locks are displayed in the standard view. `LockAct` is not set with interlock.

**Bit = 1**: `LocalSetting 2` and `4` locks are displayed in the standard view according to the interlock. `LockAct` is set according to interlock. This setting is used for hardware interlock.

**Note**

A decreasing motor protection (`Trip.Value`=0) is displayed at the output parameter `LockAct`, regardless of the `Feature` bit setting.

This `Feature` bit is also applicable for the `LocalSetting 5` in VlvS.

### 2.1.9.86 Vibrate after torque monitoring

#### Feature2 bit

Number of the `Feature2` bit: 6

#### Vibrate after torque monitoring

You can use this `Feature2` bit to determine whether vibration should be started after a torque monitoring error `TorqOpen` or `TorqClose`.

The default setting is 0.

**Bit** = **0**: No vibration.

**Bit** = **1**: Vibration is started after a torque monitoring error.

### 2.1.9.87 With acknowledge overdosage

#### Feature2 bit

Number of the `Feature2` bit: 24

#### With acknowledge overdosage

You can use this `Feature2` bit to change the acknowledgment behavior of overdosage and underdosage.

The default setting is 0.

**Bit = 0**: After an overdosage is identified, the block changes directly to the "End" state.

**Bit = 1**: After an overdosage is identified, the block changes and continues to be in the "Off" state. An acknowledgment is necessary to change the dosing state to "End". In the standard view, status display shows the state "Ack Dos End".

## 2.1.9.88 Ramp rate calculation

### Feature bit

Number of the `Feature` bit: 6 (7 for VlvAnL)

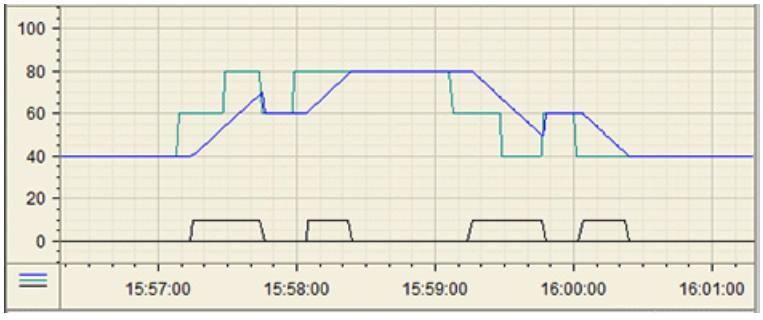
### Define ramp gradient after a change of ramp target value

You can use this `Feature` bit in the standard functions "Using setpoint ramp" or "Using a manipulated variable ramp" to define the new ramp gradient after a change of ramp target value during a ramp trip.

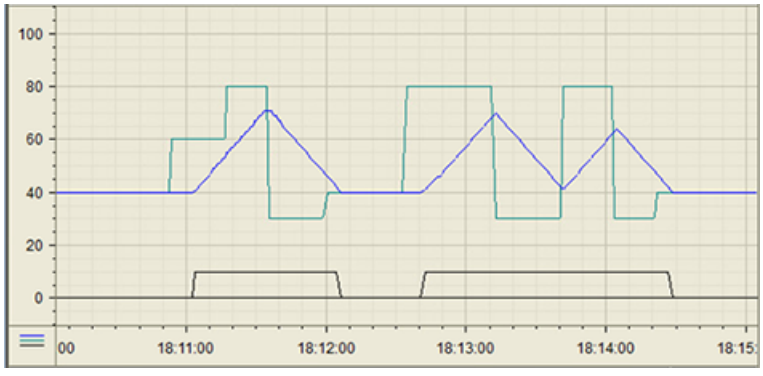
The default setting is 0.

Bit = 0:
New ramp gradient is calculated with the start point of ramp trip and the new ramp target value.

Ramp mode =Gradient:
Ramp gradient stays Constant. Ramp Trip is aborted, if new ramp target value crosses the instantaneous ramp trip value. After an abort the setpoint is tracked to the ramp target value.

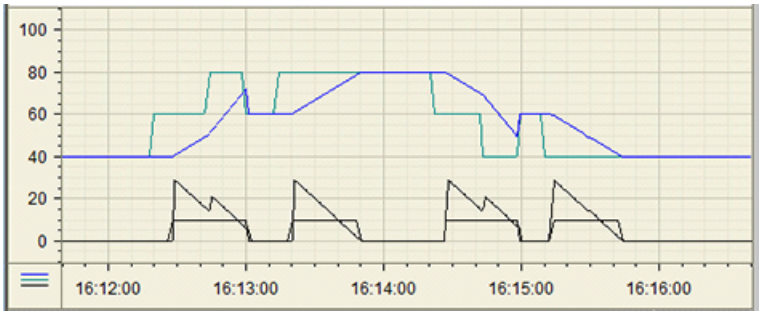



Ramp gradient changes if new ramp target value is at the other side of the start point. Ramp Trip is not aborted.




- `SP_RmpTarget`: Ramp target value
- `SP`: Setpoiont
- `SP_RmpOn`: Ramp On/Off

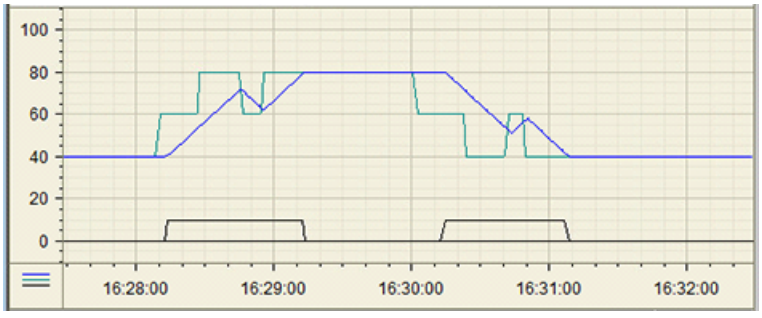
Ramp mode =Duration:
Ramp gradient will be adapted. Ramp Trip is aborted,if new ramp target value crosses the instantaneous ramp trip value. After an abort the setpoint is tracked to the ramp target value. The remaining ramp time is adapted (not reset), because the calculation of ramp gradient depends on start point and causes the increase of remaining ramp time.




- `SP_RmpTarget`: Ramp target value
- `SP`: Setpoiont
- `SP_RmpOn`: ramp On/Off
- `SP_RemRT`: Remaining ramp time

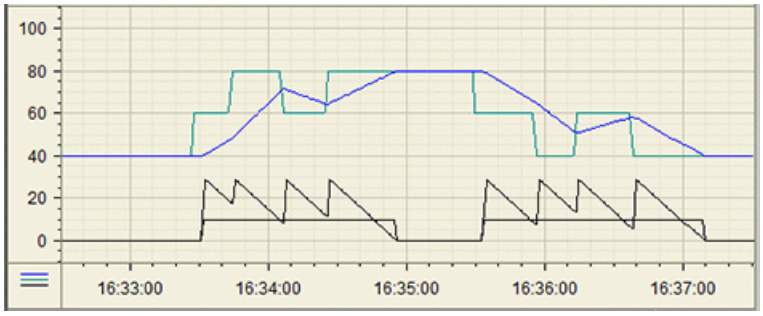
Bit = 1:
Ramp rate is calculated with the actual SP or MV and the new ramp target value.

Ramp mode =Gradient:
Ramp gradient stays Constant . Ramp Trip is not aborted,if new ramp target value crosses the instantaneous ramp trip value.




- `SP_RmpTarget`: Ramp target value
- `SP`: Setpoiont
- `SP_RmpOn`: ramp On/Off

Ramp mode =Duration:
Ramp gradient will be adapted. Ramp Trip is not aborted,if new ramp target value crosses the instantaneous ramp trip value. The ramp remaining time is completely reset with every change of ramp target value.




- SP_RmpTarget: Ramp target value
- SP: Setpoiont
- SP_RmpOn: ramp On/Off
- SP_RemRT: Remaining ramp time

### 2.1.9.89 SP following PV in open loop has no priority over SP_Ext and SP limits

#### Feature bit

Number of the Feature bit: 23

#### SP following PV in open loop has no priority over SP_Ext and SP limits

You can use this Feature bit to specify whether the setpoint follows the process value in "manual mode" with tracking (SP_TrkPV = 1). You can specify this function in case of setpoint is external or limited.

The default setting is 0.

**Bit = 0**: Setpoint follows process value (SP = PV) irrespective of whether setpoint is external or limited.

**Bit = 1**: If the setpoint is external, setpoint follows the external setpoint (SP = SP_Ext) and will be limited to the external limits (SP_ExHiLim and SP_ExLoLim). If the setpoint is internal, setpoint follows the process value (SP = PV) and will be limited to the internal limits (SP_InHiLim and SP_InLoLim).

### 2.1.9.90 With accelerated return of the integral action from the manipulated variable limit

#### Feature bit

Number of the Feature bit: 17

### With accelerated return of the integral action from the manipulated variable limit

When switching to the automatic mode (manually, by tracking, forced tracking, or program mode with manipulated variable specification) or when the controller is initialized by the `InitPid` input, the integral action (reset) is moved to preserve bumpless operation in case of an impending control deviation. The greater the control deviation and the controller gain, the greater is this shift. The result may be that the integral action is set far outside the manipulated variable limits. If the control deviation decreases after switching to the automatic mode, it may happen that in controllers with large `TI` values, the manipulated variable stays within the limit for a longer duration.

You can use this `Feature` bit to accelerate the return of the integral action from the limit. In case of a pending control deviation, the integral action is moved only in the direction of the control range to the extent that the manipulated variable can remove itself quickly and bumplessly from the manipulated variable limits (`MV_HiLim`/`MV_LoLim`).

The default setting is 0.

**Bit = 0**: Without accelerated return of the integral action.

**Bit = 1**: With accelerated return of the integral action when the integral action and the manipulated variable are outside or at the manipulated variable limits.

**Note**

In regular control mode, the integral action cannot move outside the manipulated variable limits because the manipulated variable is limited first and then fed back to the reset memory (see also anti-windup in PIDConR functions (Page 1103)).

## 2.1.9.91 External/internal selection specification

### Feature bit

Number of the `Feature` bit: 6

### External/internal selection specification

You can use this `Feature` bit to switch from the faceplate between internal and external selection of the inputs `In01` to `In10` (`SelInt` or `SelExt`).

The default setting is 0.

- **Bit = 0**:

| | |
|---|---|
| LiOp = 0 | Selection of In01 to In16 is internal; SelInt is active and can be used via faceplate. |
| LiOp = 1 | Selection of In01 to In16 is external; SelExt is active and can be used via interconnection. |

- **Bit = 1**:

| | |
|---|---|
| LiOp = 0 | With the input parameter SelIntOp or SelExtOp, you can change the selection between internal (SelInt) and external (SelExt) via faceplate. |
| LiOp = 1 | With the input parameter SelIntLi or SelExtLi, you can change the selection between internal (SelInt) and external (SelExt) via interconnection. |

### 2.1.9.92 Setting switch or button mode for local commands

#### Feature2 bit

Number of the Feature2 bit: 4

#### Setting switch or button mode for local commands (input signal as pulse signal or as static signal)

You can use this Feature2 bit to determine whether a separate interconnectable 1-active control input has to be used for every local command of the block or two local commands are assigned to a control input.

The Feature2 bit affects the following control inputs for local mode in the form of a pulse (pushbutton operation) or a static signal (switching mode):

- Starting and stopping a motor
- Opening and closing a valve

**Note**

- The input commands are checked for plausibility. For more information, see Error handling (Page 128), e.g. MotL error handling (Page 1390) and Feature2.Bit3 (Control priority in the event of invalid or not plausible input commands (Page 186)).
- With Feature2.Bit9 =1 (Button mode with triggered local control commands (Page 209)), the local input commands for button mode always need a 0->1 trigger to activate the control output signal.

You can find the commands for controlling the block in the relevant section on block operating modes. They are always the parameters that are used for the local operation of a block.

**Bit = 0**: Button mode: Each local command is assigned to a control input. This has a latching reaction and is 1-active.

**1. Example with a motor MotRevL**: In this case, use the interconnectable input parameters.

- `FwdLocal` = 1 for the command "Start forward"
- `RevLocal` = 1 for the command "Start backwards"
- `StopLocal` = 1 for the stop command

**Bit = 1**: Switching mode: two static local commands are assigned to a control input.

**2. Example with a motor MotRevL**: In this case, use the interconnectable input parameters.

- `FwdLocal` = 1 for the command "Start forward"
- `RevLocal` = 1 for the command "Start backwards"
- `FwdLocal` = 0 and `RevLocal` = 0 for the stop command

The `StopLocal` control input is irrelevant in this case.

**3. Example with motor MotSpdL**: In this case, use the interconnectable input parameters.

- `Spd1Local` = 1 for the command Motor to run with Speed1
- `Spd2Local` = 1 for the command Motor to run with Speed2
- `Spd1Local` = 0 and `Spd2Local` = 0 for the stop Command

**4. Example with motor MotSpdCL**: In this case, use the interconnectable input parameters.

- `FwdLocal` = 1 for the command "Start forward"
- `RevLocal` = 1 for the command "Start backwards"
- `FwdLocal` = 0 and `RevLocal` = 0 for the stop command

**5. Example with VlvPosL**: In this case, use the interconnectable input parameters.

- `OpenLocal` = 1, Command for Valve to "Open”
- `CloseLocal` = 1, Command for Valve to "Close”
- `OpenLocal` = 0 and `CloseLocal` = 0 for the stop command

**6. Example with Vlv2WayL**: In this case, use the interconnectable input parameters.

- `Pos1Local` = 1, Command for Valve1 to "Open"
- `Pos2Local` = 1, Command for Valve2 to "Open"
- `Pos1Local` = 0 and `Pos2Local` = 0, Command for "Position 0" (neutral position),

**7. Example with VlvL**

- `OpenLocal` = 1, Command for Valve to "Open"
- `OpenLocal` = 0, Command for Valve to "Close"

With `Feature2.Bit4` = 1, two static local commands are assigned to a control input.

### 2.1.9.93 Gradient limitation with time duration

#### Feature bit

Number of the `Feature` bit:

- 11 for PIDConL, PIDConR, and PIDStepL
- 12 for MotSpdCL, VlvAnL, and OpAnL

#### Gradient limitation with time duration

You can use this `Feature` bit to ramp up or ramp down the setpoint `SP` respectively `MV` as a function of time duration. With this `Feature` bit, the gradient limitation function can also work with the time duration.

The default setting is 0.

**Bit = 0**: Gradient limitation works only with the gradient values: `SP`/`MV_UpRaLim`, `SP`/`MV_DnRaLim`

**Bit = 1**: Gradient limitation also works with the time duration with the following parameters: `SP`/`MV_RmpModTime`, `SP`/`MV_RmpTime`.

### 2.1.9.94 Analog input 1 is reserved for the operator

#### Feature bit

Number of the `Feature` bit: 7

#### Analog input 1 is reserved for the operator

You can use this `Feature` bit to reserve the analog input 1 (`In01`) for the operator.

The default setting is 0.

**Bit = 0**: Operator cannot use the analog input 1 (`In01`).

**Bit = 1**: Operator can change the value of analog input 1 (`In01`) if the input is not connected and `OS_Perm.Bit22` is set to 1.

### 2.1.9.95 Operator can change the setpoint via faceplate also in the "Local" mode

#### Feature2 bit

Number of the `Feature2` bit: 6

#### Operator can change the setpoint via faceplate also in the "Local" mode

You can use this `Feature2` bit to enable the input of a setpoint for the operator in the "Local" mode.

The default setting is 0.

**Bit = 0**: Operator cannot change the setpoint in the "Local" mode.

**Bit = 1**: Operator can change the setpoint via faceplate in the "Local" mode.

### 2.1.9.96 Define the setpoint after stop and start of the motor

#### Feature2 bit

Number of the `Feature2` bit: 7

#### Define the setpoint after stop and start of the motor

You can use this `Feature2` bit to define the setpoint after stop and start of the motor.

The default setting is 0.

**Bit = 0**: The setpoint is retained after the motor is stopped.

**Bit = 1**: After the motor is stopped, the setpoint will be set to `SP_Off`. If the motor is started and the setpoint is in internal mode, the setpoint will be set to the low limitation `SP_LoLim`. Up to this value, the internal setpoint can be changed. In case the setpoint is external after the start of the motor, the setpoint will be changed directly to the external setpoint. The low limitation will be limited downwards to `SP_Off` and will be written back if the limitation is reached.

### 2.1.9.97 Retain last output value in case of bad input signal status

#### Feature bit

Number of the `Feature` bit: 5

#### Retain last output value in case of bad input signal status

You can use this `Feature` bit to specify that the block holds its output value if the input signal status is bad (16#00 or 16#28).

The default setting is 0.

**Bit = 0**: Default state of the block.

**Bit = 1**: Retain last output value in case of bad input signal status.

### 2.1.9.98 Forcing commands in the Local mode

#### Feature2 bit

Number of the `Feature2` bit: 8

### Forcing commands in the Local mode

You can use this `Feature2` bit to specify whether forcing commands is possible in the "Local" modes "1" and "3".

The default setting is 0.

**Bit = 0**: In the "Local" mode, forcing commands is not possible.

**Bit = 1**: In the "Local" mode, forcing commands is possible.

## 2.1.9.99 Motor stop in end position depends only on the corresponding feedback signal

### Feature bit

Number of the `Feature` bit: 15

### Motor stop in end position depends only on the corresponding feedback signal

With this `Feature` bit, you can specify whether the motor stop in end position depends only on the corresponding feedback signal or on both feedback signals.

The default setting is 0.

**Bit = 0**: Motor stop in end position depends on two feedback signals (`FbkOpen` and `FbkClose`). For example, motor stop in the end position "Open" depends on both feedback signals "Opened" and "Closed". In this case, `FbkOpen` = 1 and `FbkClose` = 0 are parameterized such that the drive output control of the motor "Open" is reset.

**Bit = 1**: Motor stop in end position depends only on the corresponding feedback signal. For example, motor stop in the end position "Open" depends only on the feedback signal "Opened". In this case, only `FbkOpen` = 1 is parameterized such that the drive output control of the motor "Open" is reset.

## 2.1.9.100 Substitution value is active if the block is in bypass

### Feature bit

Number of the `Feature` bit: 9

### Substitution value is active if the block is in bypass

You can use this `Feature` bit to specify whether the block uses the process value or a substitution value in case of an active bypass.

The default setting is 0.

**Bit = 0**: Block will use the process value (`PV`/`In`/`Out`).

**Bit = 1**: Block will use the substitution value (`BypPV`/`BypIn`/`BypOut`).

### 2.1.9.101 Enable external message

#### Feature bit

Number of the `Feature` bit: 9

#### Enable external message

You can use this `Feature` bit to specify if you want to use the quality code of `Inx`/`InTSx` or the external values `ExtValxx` as the associated value.

The default setting is 0.

**Bit = 0**: Use quality code of Inx or InTSx as the associated value.

**Bit = 1**: Use external values `ExtValxx` as the associated value.

### 2.1.9.102 Considering bad quality of automatic commands or external values

#### Feature/Feature2 bit

Number of the `Feature`/`Feature2` bit: 10

#### Considering bad quality of automatic commands or external values

You can use this `Feature/Feature2` bit to consider the bad signal status (16#00 or 16#28) in the parameter for the automatic commands or external values to move the block into the neutral position.

The default setting is 0.

**Bit = 0**: A bad signal status will not be considered.

**Bit = 1**: A bad signal status (16#00 or 16#28) in the automatic commands or external values will be considered from the block. The block goes to a defined neutral position.

### 2.1.9.103 Disable calculation of impulse control in LocalSetting 2 and 4

#### Feature bit

Number of the `Feature` bit: 20

#### Disable calculation of impulse control in `LocalSetting` 2 and 4

You can use this `Feature` bit to disable the tracking of feedback signals in the "Local" mode setting 2 and 4.

The default setting is 0.

**Bit = 0**: Calculation of impulse controls in `LocalSetting` 2 and 4.

**Bit = 1**: Disable calculation of Impulse controls in `LocalSetting` 2 and 4.

### 2.1.9.104 Inverter enable

#### Feature bit

Number of the `Feature` bit: 8

#### Inverter enable

You can use this `Feature` bit to specify whether, after the activation of `Feature.Bit15` (`Feature.Bit15` = 1), a frequency converter with a separate device infeed is to be used or the inverter enable.

The default setting is 0.

**Bit = 0**: Frequency converter with separate device infeed

**Bit = 1**: Inverter enable

### 2.1.9.105 Scale dosing can be started only if StandStill = 1

#### Feature bit

Number of the `Feature` bit: 14

#### Scale dosing can be started only if `StandStill` = 1

You can use this `Feature` bit to specify the start of scale dosing.

The default setting is 0.

**Bit = 0**: Start scale dosing independent of standstill.

**Bit = 1**: Start scale dosing only with `StandStill` = 1

### 2.1.9.106 Delete history of MV and DV moves during restart

#### Feature bit

Number of the `Feature` bit: 8

#### Delete history of MV and DV moves during restart

You can use this `Feature` bit to determine whether the history of MV and DV moves will be deleted after CPU restart or block restart.

The default setting is 0.

**Bit = 0**: The history of MV and DV will be deleted only after CPU restart if OB100 is running.

**Bit = 1**: The history of MV and DV will be deleted after CPU restart and also after a block restart (`Restart` = 1), if user DBs/FCs will be loaded in controller.

### 2.1.9.107 Status output calculation

#### Feature bit

Number of the `Feature` bit: 8

#### Status output calculation

You can use this `Feature` bit to define how the block calculates the status of output `Out`.

**Bit = 0**: The Status output is calculated with actual status input `In.ST` and the last status output `Out.ST`. The block makes a worst status calculation like technological blocks over the whole integration. With function "Tracking values" (`OutTrkOn` = 1) the status is reset to `OutTrk.ST`.

**Bit = 1**: The Status output is equal to status input (`Out.ST` = `In.ST`).

### 2.1.9.108 Suppress MsgLock and “Out of service” mode for a connected message block

#### Feature/Feature2 bit

Number of the `Feature2` bit: 11

#### Suppress `MsgLock` and “Out of service” mode for a connected message block

With this `Feature/Feature2` bit, you can suppress the message lock with the parameter `MsgLock` and the switchover to the “Out of service” mode for a connected message block (EventTs, Event16Ts). Only the connected block over the parameter `EventTsIn` is concerned. This allows generating messages also in the case of `MsgLock` = 1 or during a switchover to the "Out of Service" mode in the main block. For example, if the main block goes to the “Out of service” mode, the switchover can be messaged over a connection from the main block output `OosAct` to an input `Inx` of EventTs/Event16Ts.

The default setting is 0.

**Bit = 0**: No suppression.

**Bit = 1**: A message lock with the parameter `MsgLock` and a switchover to the “Out of service” mode are supressed for connected message block over `EventTsIn`.

### 2.1.9.109 Variants of CPI calculation

#### Feature bit

Number of the `Feature` bit: 8

### Variants of CPI calculation

With this `Feature` bit, you can choose between the two variants of CPI calculation.

The default setting is 0.

**Bit = 0**: Approximate calculation with reduced computing power consumption.

**Bit = 1**: More precise calculation corresponding to mathematical definition of variance.

**Note**

If you change the variant of calculation, the limits `CPI_AL_Lim` and `CPI_WL_Lim` may have to be adapted.

## 2.1.9.110 Scalable measuring range

### Feature bit

Number of the `Feature` bit: 6

### Scalable measuring range

With this `Feature` bit, the scalable measuring range is activated. This `Feature` bit is used by the driver generator according to the information from the HW Config / STEP 7 attribute and therefore this `Feature` bit is set by the driver wizard.

The default setting is 0.

**Bit = 0**: Scalable measuring range is not active.

**Bit = 1**: Scalable measuring range is active.

## 2.1.9.111 2 decimal points (scalable measuring range)

### Feature bit

Number of the `Feature` bit: 7

### 2 decimal points (scalable measuring range)

With this `Feature` bit, the measuring range resolution is set to 2 decimal points. This `Feature` bit is used by the driver generator according to the information from the HW Config / STEP 7 attribute and therefore this `Feature` bit is set by the driver wizard.

The default setting is 0.

**Bit = 0**: Measuring range resolution is not set to 2 decimal points.

**Bit = 1**: Measuring range resolution is set to 2 decimal points.

### 2.1.9.112 3 decimal points (scalable measuring range)

#### Feature bit

Number of the `Feature` bit: 8

#### 3 decimal points (scalable measuring range)

With this `Feature` bit, the measuring range resolution is set to 3 decimal points. This `Feature` bit is used by the driver generator according to the information from the HW Config / STEP 7 attribute and therefore this `Feature` bit is set by the driver wizard.

The default setting is 0.

**Bit = 0**: Measuring range resolution is not set to 3 decimal points.

**Bit = 1**: Measuring range resolution is set to 3 decimal points.

### 2.1.9.113 Output substitute values in case of communications error

#### Feature bit

Number of the `Feature` bit: 29

#### Output substitute values in case of communications error

Use this `Feature` bit to specify if substitute values are to be outputted in case of communication error.

The default setting is 0.

**Bit = 0**: Substitute values are not outputted in case of communication error.

**Bit = 1**: Substitute values are outputted in case of communications error

### 2.1.9.114 Output last values in case of communications error

#### Feature bit

Number of the `Feature` bit: 29

#### Output last values in case of communications error

Use this `Feature` bit to specify if last values are to be outputted in case of communication error.

The default setting is 0.

**Bit = 0**: Last values are not outputted in case of communication error.

**Bit = 1**: Last values are outputted in case of communications error.

### 2.1.9.115 Dosing controls are deactivated during scale is in taring mode

#### Feature bit

Number of the `Feature` bit: 14

#### The dosing controls are deactivated during scale is in taring mode

You can specify the dosing controls `Ctrl`, `P_Ctrl`, `Ctrl2`, `P_Ctrl2` after the start of scale dosing using this `Feature` bit.

The default setting is 0.

**Bit = 0**: After start of scale dosing the controls are active and independent of `StandStill`.

**Bit = 1**: After start of scale dosing the controls are deactivated during scale is in taring mode (`StandStill` = 0).

**Note**

After start of scale dosing during taring mode (`StandStill` = 0) the display area in the standard view shows "Taring".

### 2.1.9.116 Do not evaluate the first run of the diagnostic blocks

#### Feature bit

Number of the `Feature` bit: 27

#### Do not evaluate the first run of the diagnostic blocks

Use this `Feature` bit to deactivate the evaluation of the first run detection in parameter `DXCHG1` for channel blocks in case of manually configuration of the parameter `Mode`.

The default setting is 0.

**Bit = 0:** Activates the evaluation of the first run detection in parameter `DXCHG1`.

**Bit = 1:** Deactivates the evaluation of the first run detection in parameter `DXCHG1`.

### 2.1.9.117 Selection of metadata and diagnostic blocks

#### Feature bit

Number of the `Feature` bit: 9

### Selection of metadata and diagnostic blocks

Use this `Feature` bit to specify how PCS 7 driver generator proceeds with the connected hardware.

The default setting is 0.

**Bit = 0:** PCS 7 driver generator uses PCS 7 Basis Library metadata and function blocks.

**Bit = 1:** PCS 7 driver generator uses metadata and function blocks from an additionally installed AddOn Library.

## 2.1.9.118 Limits disabled in faceplate

### Feature bit

Number of the `Feature` bit: 23

Blocks: CntOhSc, CountOh, CountScL, TotalL, MonAnL, MonAnS, MotL, MotRevL, MotSpdCL, MotSpdL, VlvAnL, VlvMotL, and VlvPosL

Number of the `Feature` bit: 16

Blocks: DoseL

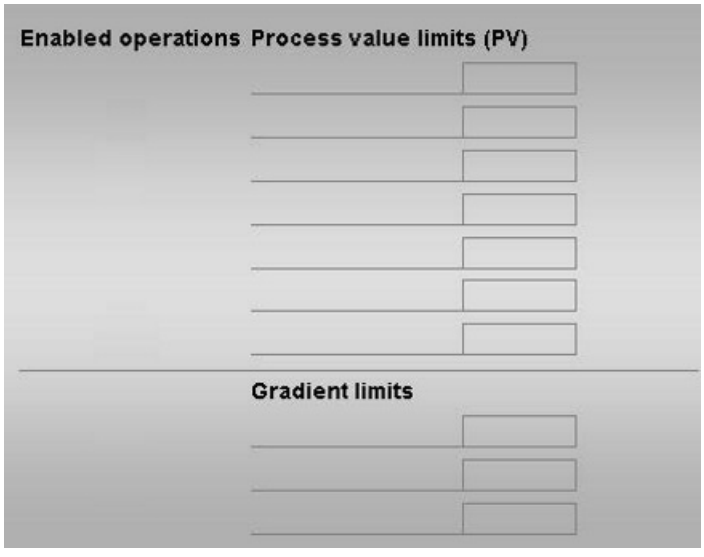
Number of the `Feature` bit: 31

Blocks: FmCont, FmTemp, PIDConL, PIDConR, PIDConS, PIDStepL, and ConPerMon

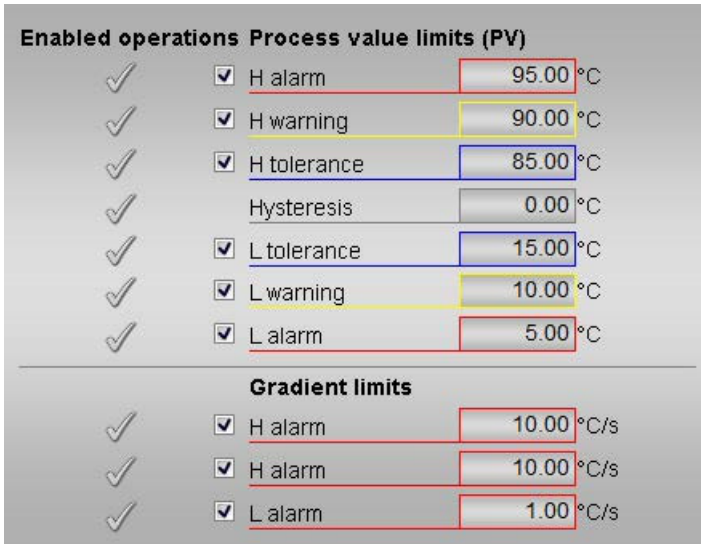
### Limits disabled in faceplate

You can use this `Feature` bit to make the limits visible but disabled in faceplate when limits are disabled.

The default setting is 0.




**Bit** = **0**: Limits are invisible in faceplate when disabled from block.




**Bit** = **1**: Limits are visible but disabled in faceplate when disabled from block.

### 2.1.9.119 Keep last valid raw value if input value is bad

#### Feature bit

Number of the `Feature` bit: 31

#### Keep last valid raw value if input value is bad

Use this `Feature` bit to hold the raw output value if the input value is bad (16#00 or 16#28).

The default setting is 0.

**Bit = 0:** Transfers the data. Status of input value is not bad.

**Bit = 1:** Does not transfer the data. Status of input value is bad.

### 2.1.9.120 With PID initialization if mode SP intern extern changes

#### Feature bit

Number of the `Feature` bit: 14

#### With PID initialization if mode SP intern extern change

You can use this `Feature` bit to define the PID behavior after a change of mode SP intern/extern (`SP_ExtAct`). This means MV makes a jump or not.

The default setting is 0.

**Bit = 0:** No PID initialization after a change of mode SP intern/extern (`SP_ExtAct`). If the setpoint intern is different to SP extern, MV makes a jump to reach the new setpoint faster.

**Bit = 1**: With PID initialization after a change of mode SP intern/extern (`SP_ExtAct`). MV makes no jump, but the process needs more time to reach the new setpoint.

### 2.1.9.121 Changes signal status of outputs in OOS

#### Feature/Feature2 bit

Number of the `Feature` bit:

- 3 for SelA16In, SelAn16L
- 4 for CntOhSc, CountScL, CountOh, TotalL
- 5 for PIDConS, PIDConR, PIDConL, PIDStepL, FmTemp, FmCont, MPC 10x10, KalFit, ModPreCon, TimeTrig, AV, MonAnL, MonAnS, MonDiL, MonDi08, MonDiS

Number of the `Feature2` bit:

- 1 for MotL, MotS, MotRevL, MotSpdL, MotSpdCL, VlvS, VlvL, Vlv2WayL, VlvMotL, VlvAnL, VlvPosL
- 6 for DoseL

Refer the section "Out of service mode" of the respective blocks to know about the applicable outputs.

#### Changes signal status of outputs in OOS

This `Feature` bit changes the signal status of outputs in Out of service mode.

The default setting is 0.

Bit = 0: Default state of the block.

Bit = 1: Signal status of the respective output parameters changes to 16#60.

**Note**

Number of the `Feature` bit is different for various families of the blocks. The `Feature` "Changes signal status of outputs in OOS" has higher priority over the `Feature` "Reaction of the switching points in the "Out of service" operating mode".

### 2.1.9.122 Behavior in the case of Bus failure

#### Feature bit

Number of the `Feature` bit: 7

#### Behavior in the case of Bus failure

With this `Feature` bit, you can set the signal status of all output signals to 16#68 in the case of Bus failure (`ModErr` = 1).

The default setting is 0.

**Bit = 0**: The signal status of all output signals is set to 16#00 ("Bad, device related").

**Bit = 1**: The signal status of all output signals is set to 16#68 ("Uncertain, device related").

### 2.1.9.123 Button mode with triggered local control commands

#### `Feature2` bit

Number of the `Feature2` bit: 9

#### Button mode with triggered local control commands

You can use this to change the behavior of local mode with local commands (LocalSetting =1 or 3). The precondition is - Button mode: `Feature2.Bit4` = 0.

**Bit = 0**: Button mode with static input commands - After a plausibility control, the local input commands are active.

**Bit = 1**: Button mode with triggered input commands - The local input commands always trigger a 0->1 switching before the control output signal is active. If a block changes into local mode from another mode or to an error state, then a trigger is necessary.

### 2.1.9.124 Understeering / underflow for measuring range 0...10V

#### `Feature` bit

Number of the `Feature` bit: 11

#### Activate understeering / underflow for measuring range 0...10V

You can use this `Feature` bit to activate understeering / underflow handling for measuring range 0...10V.

The default setting is 0.

Bit = 0: understeering / underflow handling for measuring range 0...10V - OFF (default).

Bit = 1: understeering / underflow handling for measuring range 0...10V - ON.

### 2.1.9.125 Disable limit act output when limit value message is disabled

#### Feature/Feature2 bit

Number of the `Feature2` bit: 12

Number of the `Feature` bit: 3

### Disable limit act output when limit value message is disabled

You can use the `Feature/Feature2` bit to disable the limit value act output (`xx_xx_Act`) when the limit value message (`xx_xx_MsgEn`) is disabled.

The default setting is 0.

**Bit = 0**: Limit active output is not disabled when limit value message is disabled

**Bit = 1**: Limit active output is disabled when limit value message is disabled

## 2.1.9.126 Resetting the Pending Error in Local Mode 1 and 3

### Feature bit

Number of the `Feature2` Bit: 13

Resetting Pending Error in Local Mode 1 & 3.

With this `Feature` bit, you define you can define whether the Pending Errors can be resetted by using the faceplate/RstLi or not.

The default setting is 0.

**Bit = 0**: Resetting of pending errors cannot be possible via faceplate/RstLi.

**Bit = 1**: Resetting of pending errors can be possible via faceplate/RstLi.

## 2.1.9.127 Timestamp calculation

### Feature bit

Number of the Feature bit: 5

### Timestamp calculation

The calculation of the time stamp `TS_Out` has been improved. You can select via `Feature.Bit5` whether the CPU time or the IO time is used.

- Bit5=0: IO time stamp (default setting)
- Bit5=1: CPU time stamp

**Note**

Modifications are done in the Pcs7DiIT block prevent invalid time stamping error messages.

### 2.1.9.128 Reversal of control direction for fail-open valves

#### Feature bit

Number of the Feature bit: 5

#### Reversal of control direction for fail-open valves

`Feature.Bit5` = 1 turns around the control direction of scaled input `PV_In.Value` or `SimPV_In.Value`.

Example, with `Scale.Low` = 0 and `Scale.High` = 100 :

- With `Feature.Bit5` = 1, `PV_In.Value` = 0 leads to `PV_Out.Value` = 16#6C00.
- With `Feature.Bit5` = 1, `PV_In.Value` = 100 leads to `PV_Out.Value` = 0.
- With `Feature.Bit5` = 0, `PV_In.Value` = 0 leads to `PV_Out.Value` = 0.
- With `Feature.Bit5` = 0, `PV_In.Value` = 100 leads to `PV_Out.Value` = 16#6C00.

### 2.1.9.129 External Input to enable or Disable

#### Feature bit

Number of `Feature` Bit: 6

External Input to enable or Disable.

With this function you can activate External input.

Default = 0.

Bit = 0: External input disabled.

Bit = 1: External input enabled.

### 2.1.9.130 Switch over to internal time enable or Disable

#### Feature bit

Number of `Feature` Bit: 7

Switch over to internal time enable or Disable.

With this function you can switch to Internal time if the signal status of the external signal is bad.

Default = 0.

Bit = 0: internal switch over disabled.

Bit = 1: internal switch over enabled.

### 2.1.9.131 AS time enable or Disable

**Feature bit**

Number of `Feature` Bit: 8

AS time enable or Disable.

With this function you can activate AS Time or the CPU time.

Default = 0.

Bit = 0: AS time disabled.

Bit = 1: AS time enabled.

### 2.1.9.132 Use connectable inputs OpenLi, StopLi, CloseLi with CtrlLiOp

**Feature bit**

Number of the `Feature` bit: 7

Use connectable inputs `OpenLi`, `StopLi`, `CloseLi` with `CtrlLiOp`

You can use this `Feature` bit to specify whether the block uses the inputs `OpenLi`, `StopLi`, `CloseLi` as a function of `CtrlLiOp` or `ModLiOp`.

The default setting is 0.

Bit = 0: `OpenLi`, `StopLi`, `CloseLi` are used as a function of `ModLiOp`.

Bit = 1: `OpenLi`, `StopLi`, `CloseLi` are used as a function of `CtrlLiOp`.

### 2.1.9.133 Selecting the Feature Bit when Namur Off = 1

**Feature bit**

Number of the `Feature` bit: 12

With `Feature.Bit12` = 1 and `NamurOff` = 1

- Exceeding of the limit defined at input parameter HighLimit, leads to `PV_HiAct`.Value=1.
- Undershooting of the limit defined at input parameter LowLimit, leads to `PV_LoAct`.Value=1.

Regardless `Feature.Bit`12, `PV_HiAct`.Value and `PV_LoAct`.Value will be set in case of nominal range is left.

### 2.1.9.134 Bypass over both connection BypLix OR Inx

**Feature bit**

Number of the `Feature` bit: 3

With this Feature bit you can use both connected bypass information BypLix, Inx to bypass an interlock input.

The default setting is 0

**Bit3 = 0:** BypLix.Value is used to bypass an interlock input.

**Prerequisite:** BypLix <> 16#FF

With a change of BypLix.Value the bypass of an interlock input BypInx is set to BypLix.Value.

**Bit3 = 1:** BypLix.Value, Inx.Bit1 are used to bypass interlock inputs.

**Prerequisite:** BypLix <> 16#FF, Feature Bit2 =1, BypEn.Inx = 1

With a change of BypLix.Value, Inx.Bit1 the bypass of an interlock input BypInx is set to "BypLix.Value OR Inx.Bit1". Both inputs are false to reset the bypass. After that operator can change bypass over faceplate.

With changes of BypLix.Value or Inx.Bit1 the input BypInx is tracked. After that operator can change again bypass over faceplate.

### 2.1.9.135 Resynchronization after startup

#### Feature/Feature2 Bit

Number of the Feature bit: 17

**Drives (**Feature2bit17) **:** MotL, MotRevL, MotSpdL, MotSpdCL, VlvL, VlvMotL, VlvAnL, VlvPosL

**Operate blocks (**Featurebit17)**:** OpDi01, OpDi03, OpAnL

You can use the Feature bit "**Resynchronization after startup**" to track controls and block state after CPU startup through feedback signals and readback value.

The default setting is 0.

**Bit = 0:** Startup without Resynchronization

**Bit = 1:** Startup with Resynchronization: The control outputs and block state are tracked through the feedback signals and readback value.

### 2.1.9.136 Resynchronization after out of service or after simulation

#### Feature/Feature2 Bit

Number of the Feature bit: 18

**Drives:** MotL, MotRevL, MotSpdL, MotSpdCL, VlvL, VlvMotL, VlvAnL, VlvPosL
After leaving "Out of service" mode or after leaving simulation function you can use the Feature2 bit to track controls and block state through feedback signals and readback value.

**Operate blocks:** OpDi01, OpDi03, OpAnL
After leaving "Out of service" mode you can use the Feature bit to track controls and block state through feedback signals and readback value.

The default setting is 0.

**Bit = 0:** Without Resynchronization

**Bit = 1:** Resynchronization after leaving "Out of service" mode or after leaving simulation function.

### 2.1.9.137 Resynchronization after CSF

#### Feature2 Bit

Number of the `Feature` bit: 19

**Drives:** MotL, MotRevL, MotSpdL, MotSpdCL, VlvL, VlvMotL, VlvAnL, VlvPosL

After a control system fault (input CSF 1->0) you can use the `Feature2` bit to track controls and the block state through feedback signals and readback value.

The default setting is 0.

**Bit = 0:** Without Resynchronization

**Bit = 1:** Resynchronization after a control system fault

### 2.1.9.138 Resynchronization with StateTrkOn input

#### Feature/Feature2 Bit

Number of the `Feature` bit: 20

**Drives (**`Feature2`bit20) **:** MotL, MotRevL, MotSpdL, MotSpdCL, VlvL, VlvMotL, VlvAnL, VlvPosL

**Operate blocks (**`Feature`bit20)**:** OpDi01, OpDi03, OpAnL

After a trigger (0->1) at StateTrkOn input, you can use the feature bit to track controls and the block state through feedback signals and readback value.

The default setting is 0.

**Bit = 0:** Without Resynchronization

**Bit = 1:** Resynchronization after a trigger with StateTrkOn input

### 2.1.9.139 Control with output StopCtrl and state “No Cmd”

#### Feature2 Bit 21

Feature2 bit: MotL, MotRevL, MotSpdL, MotSpdCL, VlvL

Number of the feature bit: 21

**Control with output StopCtrl and state "No Cmd"**

You can use this feature bit if the control outputs together with StopCtrl should be able to realize the "No Cmd" mode. In this case all control outputs are reset.

The default setting is 0.

Bit = 0: Control position "No Cmd" is not used.

Bit = 1: The control outputs with StopCtrl are able to use the control position "No Cmd".

## Using StopCtrl with control position "No Cmd"

Some actors or sometimes a subordinate building block structure need for every position a separate static control output. The following motors and valves have the control output StopCtrl to realize the stop control separately: MotL, MotRevL, MotSpCL, MotSpdL, VlvL

Normally the value of the output StopCtrl is the opposite of the control outputs which activate the drive.

| | |
|---|---|
| MotL | StopCtrl = NOT Start |
| MotRevL | StopCtrl = NOT Fwd AND NOT Rev |
| MotSpdCL | StopCtrl = NOT Fwd AND NOT Rev |
| MotSpdL | StopCtrl = NOT Spd1 AND NOT Spd2 |
| VlvL | StopCtrl = NOT Ctrl |

Sometimes after a resynchronization it is not possible to see in which position and condition the valve or the motor is, because the feedback signals are not plausible. In this case you have the opportunity with Feature2 Bit21 Control with output StopCtrl and state “No Cmd” to parameterize that the drive can control the position/state “No Cmd”.

| | Controls “No Cmd” | Not plausible combinations |
|---|---|---|
| MotL | StopCtrl =0, Start =0 | FbkStart =0, FbkStop =0,<br>FbkStart =1, FbkStop =1 |
| MotRevL | StopCtrl = 0, Fwd =0, Rev =0 | FbkFwd =0, FbkRev =0, FbkStop =0<br>FbkFwd =1, FbkRev =1, FbkStop =1<br><br>FbkFwd =1, FbkRev =1, FbkStop =0<br>FbkFwd =0, FbkRev =1, FbkStop =1<br>FbkFwd =1, FbkRev =0, FbkStop =1 |
| MotSpdCL | StopCtrl = 0, Fwd =0, Rev =0 | See MotRevL |
| MotSpdL | StopCtrl = 0, Spd1 =0, Spd2 =0 | FbkSpd1 =0, FbkSpd2 =0, FbkStop =0<br>FbkSpd1 =1, FbkSpd2 =1, FbkStop =1<br><br>FbkSpd1 =1, FbkSpd2 =1, FbkStop =0<br>FbkSpd1 =0, FbkSpd2 =1, FbkStop =1<br>FbkSpd1 =1, FbkSpd2 =0, FbkStop =1 |
| VlvL | StopCtrl =0, Ctrl =0 | FbkOpen =0, FbkClose =0,<br>FbkOpen =1, FbkClose =1 |

## 2.1.10 Functions for controllers

### 2.1.10.1 Delay alarm for control deviation at setpoint step changes

#### Alarm delay for blocks with the function "Delay alarm for control deviation at setpoint step changes"

This type of alarm delay is used when temporary violations of set alarm thresholds of the control deviation are to be suppressed at setpoint step changes. The alarm delay is parameterized at the following inputs:

| Parameter for the delay time | Explanation |
|---|---|
| ER_AH_DFac | Delay factor at positive setpoint step changes for incoming alarms at the control deviation monitoring ER_AH_Lim |
| ER_AL_DFac | Delay factor at negative setpoint step changes for incoming alarms at the control deviation monitoring ER_AL_Lim |

The effective delay time is calculated from the delay factor and the setpoint difference and is specified at the output parameter:

| Parameter | Explanation |
|---|---|
| ER_A_DCOut | Effective delay time at setpoint step changes for incoming alarms during control deviation monitoring |

#### Calculation of the output parameter ER_A_DCOut

ER_A_DC is assigned by default to the output before a setpoint change.

ER_A_DCOut = ER_A_DC
In the case of a setpoint change in the positive direction during automatic mode, the output is calculated as follows:

ER_A_DCOut = Maximum (ER_A_DC, ER_AH_DFac * Setpoint difference)

In the case of a setpoint change in the negative direction during automatic mode, the output is calculated as follows:

ER_A_DCOut = Maximum (ER_A_DC, -1*ER_AL_DFac * Setpoint difference)

When the loop control has stabilized again, meaning (ER_AL_Lim + ER_Hyst) <= ER <= (ER_AH_Lim – ER_Hyst), and the delay time for outgoing alarms ER_A_DG has expired, the output is reset again to ER_A_DC: ER_A_DCOut = ER_A_DC.

**Note**

After a setpoint jump the function "**Delay alarm for control deviation at setpoint step changes**" always only affects the limit to the direction of the setpoint jump and not to the opposite limit. If the deviation value ER reaches the opposite limit (loop is overshooting), the switch-on delay of the opposite side acts ER_A_DCOut =ER_A_DC, ER_AH_DC.

### Activating the alarm delay

The alarm delay factors have a default value of 0, meaning that the function is deactivated. Specify a delay factor > 0 in order to use the function.

### Pending alarms

Pending alarms are output at the output parameters `ER_AL_Act` and `ER_AH_Act`.

### Activating and deactivating the function

The function is deactivated (default) when `ER_AH_DFac` = 0.0 and `ER_AL_DFac` = 0.0.

## 2.1.10.2 Inverting control direction

### Inverting control direction

For some processes (for example cooling processes), negative control gain is necessary. This is achieved by the inverting the control direction by means of the input parameter `NegGain = 1`. The gain is always entered positively at the input parameter `Gain` . If there is an inversion, this is indicated at the output parameter `GainEff` by a negative number.

## 2.1.10.3 Error signal generation and dead band

### Error signal generation and dead band

The signal error is formed from the effective setpoint `SP` and the process value `PV` and is available at the `ER` output.The `ER` Output will be calculated as follows:

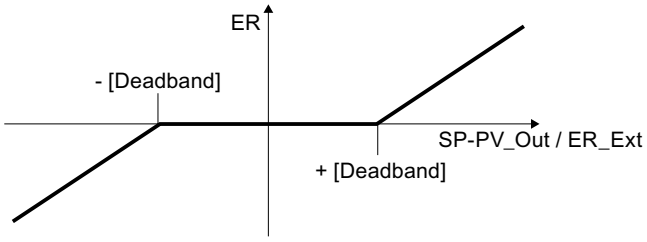
1. Without `Deadband`
   `ER = SP - PV_Out`
   PIDConR (`ER =PV_Out - SP`)
2. With `Deadband`
   `ER = SP - (PV - Deadband)`

In the case of activated error signal generation (only at block PIDConL, `Feature.Bit14`), `ER` is formed by `ER_Ext`.

To suppress disturbances in the steady state, you can assign a dead band `(Deadband)`:

- `Deadband = 0`: Dead band is disabled
- `Deadband ≠ 0`: Dead band is enabled

With a negative dead band (`Deadband < 0.0`), calculation is continued internally with this value




A dead zone in the controller can help reduce the movements of the actuator and thus reduce the energy consumption and wear of the actuator. This applies especially to mechanical actuators, such as valves and pumps. The dead zone may also required in order to avoid small, stable, continuous oscillations (working movement), which are caused by the quantization of the control actions in the control loop.

**Note**

Calming the control signals via dead band is only possible when no structure splitting is active, i.e. when `PropFacSP` = 1 and `DiffToFbk` = 0. Controller actions in the feedback branch directly process the measured actual value instead of the control deviation and are therefore not affected by the dead band.

- Remaining control deviations smaller than the dead band are ignored by the controller, i.e. it does nothing even it can. Therefore, steady states of the actual value can occur, the average time of which significantly differs from the setpoint.
- When such steady states occur at the edges of the dead zone, it must be expected that even the smallest problems will cause control intervention (i.e. actuator energy consumption).
- Large dead zones have a negative effect on the control response of the control loop for setpoint changes. This is because the controller "stops work", for example, in the rising phase of a positive step response when entering the dead band. This can lead to creeping settling time, or cause the controller to be activated again when the actual value exceeds the high limit of the dead band in the form of an overshoot.
- After a fault event that leads to exiting the dead band, the controller brings the actual value back only to the edge of the dead band, which has the above-mentioned disadvantages.

By setting `Feature.Bit30`, you can activate dynamic adjustment of the dead band. This allows you to benefit from the advantages of a dead band without having to tolerate the above-mentioned disadvantages.

When large control deviations occur (i.e. the dead band is exited), it is temporarily deactivate until the controller returns the actual value to the proximity of the setpoint. The controller also demonstrates control response for setpoint changes without a dead band in this way.

There is then a wait until it can be assumed that the process has settled on operating point. Only then is the dead band activated again. When it is reactivated, the manipulated variable is set to the mean of a sliding time window. This ensures that the actual value to be controlled settles more in the middle of the dead band than at the edges. This avoids unnecessary shifting of the mean value of the controlled variable from the setpoint, and thus poor stationary control performance. The probability that the controlled variable is within the dead zone time is

increased, and further control actions are no longer necessary. This leads to reduced wear and less energy consumption.

**Note**

If `Feature.Bit30` = 1 is set, the ConPerMon block has to be connected with the controller. The blocks PIDConL or PIDConR monitor this by checking the status signal of `MeanMV.ST`. In case of `Feature.Bit30` = 1 and `MeanMV.ST` = 16#78 the dynamic deactivation of deadband is not possible and the output `ErrorNum` will be set to 52.

## Configuration and commissioning

- Use a process tag type that contains the ConPerMon function block for monitoring the control performance in addition to the controller.
- Check whether the additional interconnections between the monitoring block and controller are available:
  - `PIDConL.MV_Mean:= ConPerMon.MV_Mean`
  - `PIDConL.SettliTime:= ConPerMon.SettliDeadBand`
- Put the controller into operation as usual with the PID Tuner, and then initialize the ConPerMon block.
- Perform a setpoint step with activated control performance monitoring and without dead band (`PIDConL.DeadBand:=0`). Various characteristics of the step response can be determined by the ConPerMon block in this way. Check if the determined settling time at the `ConPerMon.SettliTime` output parameter appears plausible.
- Specify the width of the dead band at the `PIDConL.DeadBand` input parameter.
- Check the actuating signal signal in the steady state of the control loop. Has the desired calming of the signal form been achieved?
- Activate `Feature.Bit30`.
- Check the control response of the control loop. If the control loop has not sufficiently settled as expected near the center of the dead band after a setpoint change, it may help to increase the value of the `PIDConL.SettliTime` input parameter.

### Notes on selecting the width of the dead band

The width of the dead band primarily depends on the desired precision of the control, in other words, after the maximum, permanent, control deviations permissible by process engineering and after the change of the manipulated variable based on a minimum possible change of the manipulated variable (caused, for example, by a switching actuator). With regard to the minimization of control interventions, the following adjustment rules help to select the dead band wide enough that unavoidable variances of the control variables due to measurement noise or quantization noise do not lead to permanent control movements:

- Assuming a normal distribution of the statistical control variables around the setpoint, two to three times the standard deviation of the actual value in the steady state is used as the width of the dead band. This information can be obtained from the control loop monitoring block: `PIDConL.DeadBand:= (2...3)*ConPerMon.RefStdDev`.
- If working movements in the control loop are systematically caused by the manipulated variable quantization, for example, due to a pulse width modulation with a defined minimum pulse duration and pulse period, a step controller with defined minimum step size or an electro-pneumatic positioner, the width of the dead band depends on the manipulated variable quantization multiplied by the process gain.
  **Example**: A control signal resolution of 5% for temperature regulation with the process gain 1.5°C/% means that only temperatures in a grid of 5%, 1.5°C/% = 7.5°C can be reached exactly. The dead band must then be selected wide enough sp that at least one grid point falls within the dead band in which the process can remain in a steady-state, i.e. DeadBand > 3.8°C in this case.

## 2.1.10.4 Using control zones

### Using control zones

The control zone function is mainly used for temperature processes in which the time lag is small compared to the recovery time (no more than a tenth of the recovery time). This is typically the case when the temperature controller accesses the actuator directly (e.g. heating, steam valve), and typically not the case with master controllers of cascade interconnections (e.g. control of the internal temperature of a reactor via the shell temperature).

The control zone can result in large overshoots in processes with an excessively large time lag.

During commissioning with the PCS 7 PID-Tuner, you can check whether the process order is smaller than 2 and the process type "VZ2 model" is displayed. An initial approach for dimensioning the control zone: `ConZone= MV_HiLim/Gain`.

If `ConZone > 0`, the controller works with a control zone; if `ConZone <= 0`, the "control zone" function is deactivated. With a negative control zone, an error ID is output at the

ErrorNum parameter. When the control zone is enabled, the controller operates according to the following algorithm:

- If the process value PV exceeds the setpoint SP by more than ConZone, the value MV_LoLim is output as the manipulated variable (controlled mode, in order to return to the control zone as fast as possible).
- If the process value PV falls below the setpoint SP by more than ConZone, MV_HiLim is output (controlled mode).
- (Only possible with PIDConL, not with FmTemp) In controlled mode of the control zone and when the I component is enabled (TI<>0), the I component can be set in a variety of ways when the control zone is entered, depending on three Feature bits:

| Feature.Bit12 | Feature.Bit13 | Feature.Bit18 | Response |
|---|---|---|---|
| 0 | 0 | 0 | I_Part := MV.Value -P_Part -FFwd ; The controller operates bumplessly when entering automatic control mode. (This corresponds to the reaction up to V9.0) |
| 0 | 0 | 1 | I_Part := MV.Value -FFwd ; A P jump of the manipulated variable occurs when entering automatic control mode (use only in exceptional cases) |
| 1 | x | x | I_Part := MV_Offset.Value; The controller starts with the default value MV_Offset as I component when entering automatic control mode |
| 0 | 1 | x | I component is frozen. When it enters the control zone, the controller starts with the value of the I component that was in effect when it left the control zone. This can be useful for continuous processes, where it can be assumed that the controller is at the same operating point as it was when it left the control zone, nearly in a steady state. |

x = any

The I component is always set to MV_Offset when the I component is disabled (TI=0). I.e.: The control zone operates with the specified I_Part := MV_Offset with P or PD controlling.

Freezing of the I component with IntHoldPos or IntHoldNeg has no effect during controlled mode due to the control zone.

- If the process value PV stays within the control zone (ConZone), the manipulated variable assumes the value of the PID algorithm (automatic closed-loop mode).

**Note**

The change from controlled closed-loop mode to automatic closed-loop mode is based on a hysteresis of 20% of the control zone. Make sure that the control zone has an adequate width before you manually activate the control zone. An insufficient width of the control zone leads to oscillation of the manipulated variable and of the process value.

Advantages of the control zone:

Once the control zone is entered, the manipulated variable is quickly reduced by the applied D component. The control zone is therefore only useful if the derivative component is activated. Without the derivative component, only the reducing proportional component would reduce the manipulated variable to any significant degree. The control zone speeds up settling without

overshoot or undershoot if the value of the output minimum or maximum manipulated variable is a long way from the stationary manipulated variable required for the new operating point.

### 2.1.10.5 Setpoint limiting for external setpoints

#### Setpoint limiting for external setpoints

With this function you can limit the external setpoint to a range by means of the parameters `SP_ExHiLim` (high limit) and `SP_ExLoLim` (low limit). If the setpoint lies outside the range defined by you, it is limited to the valid range.

If the external setpoint lies on or above the limit `SP_ExHiLim`, this is displayed at the output `SP_ExHiAct = 1`.

If the external setpoint lies on or below the limit of `SP_ExLoLim`, this is displayed at the output `SP_ExLoAct = 1`.

The external setpoint limits can be tracked internally via the interconnection of the output parameters `SP_InHiOut` or `SP_InLoOut` following `SP_ExHiLim` or `SP_ExLoLim`. You can control the two limit pairs from the faceplate.

### 2.1.10.6 Tracking setpoint in manual mode

#### Tracking setpoint in manual mode

To allow a bumpless transfer to automatic mode, the setpoint tracks the process value. When tracking,(`SP_TrkPV = 1`) the internal setpoint `SP_Int` is tracked to the process value `PV`.

Additional information on setpoint tracking is available in Manual and automatic mode for control blocks (Page 79).

### 2.1.10.7 Tracking and limiting a manipulated variable

#### Manipulated variable tracking

You adjust the manipulated value (tracking) in order to implement a Bumpless switchover of controllers. A typical use case is cascade control: If the assigned secondary controller is no longer in automatic mode with external setpoint, the primary controller must track it.

To adjust the manipulated variable (tracking), you have to set the parameter `MV_TrkOn = 1`. Now the manipulated variable is taken from the interconnected tracking value `MV_Trk` and passed to the output `MV`. The MV output is limited to the `MV_HiLim` and `MV_LoLim` parameters.

Manual mode takes priority over tracking to allow a plant operator to set the controller to manual mode using the faceplate even when tracking the manipulated variable and therefore continue to normal operation.

The text "Tracking" is also displayed in the standard view of the faceplate.

### Activating forced tracking mode for the manipulated variable

Forced tracking is used to set the controller output of a higher-level controller to a value that can be specified.

You can use forced tracking, for example, to implement a centralized emergency stop in the plant. This can be put into effect regardless of the operating mode the controller is currently in.

To force adjustment of the manipulated variable (tracking), you have to set the parameter `MV_ForOn = 1`. Now the manipulated variable is taken from the interconnected tracking value `MV_Forced` and passed to the output `MV`.

With forced tracking, it is not possible to limit the manipulated variable, nor can the plant operator change to manual mode in the faceplate. The text "Forced tracking" is also displayed in the standard view of the faceplate.

**Note**

This function is not available for the FmCont, FmTemp, and ModPreCon blocks.

### Limiting the value of a manipulated variable in automatic mode

In automatic mode, the manipulated variable is set to its automatic manipulated variable limits as specified by the input parameters `MV_HiLim` and `MV_LoLim` and output at the output parameter `MV`. Reaching the limit is then displayed at the output parameter `MV_HiAct = 1` for the high limit and `MV_LoAct = 1` for the low limit.

Interconnecting the output parameter `ManHiOut` or `ManLoOut` to `MV_HiLim` or `MV_LoLim` allows the automatic manipulated variable limits to tracked to manual manipulated variable limits. You can keep both limit pairs in synch and control the manual manipulated variable limits in the faceplate.

### Limiting the value of a manipulated variable in manual mode

In manual mode, the manipulated variable is set to its manual manipulated variable limits as specified by the input parameters `ManHiLim` and `ManLoLim` and output at the output parameter `MV`.

### See also

Forcing operating modes (Page 45)

## 2.1.10.8 Feedforwarding and limiting disturbance variables

### Feedforward control and limitation

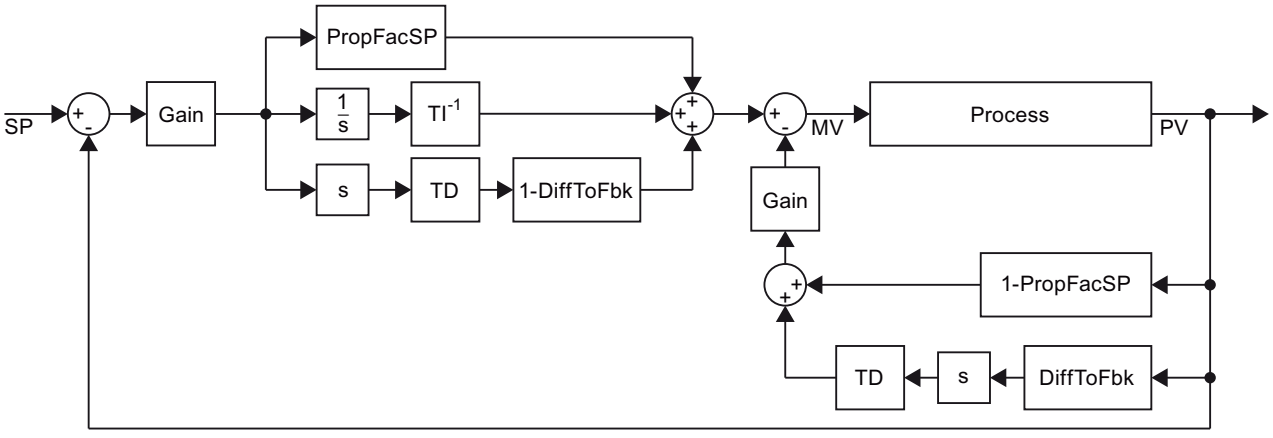
The feedforward control is used in order to compensate measurable disturbance variables, such as temperature or pressure, that can have an effect on the process. In automatic mode, the disturbance variable is added to the result of the PID algorithm.

The disturbance variable is connected to the `FFwd` Parameter. It is limited to the `FFwdHiLim` and `FFwdLoLim` limits. If the disturbance variable is outside or at the limits, this is indicated by the `FFwdHiAct = 1` or `FFwdLoAct = 1` output parameters.

### 2.1.10.9 Structure segmentation at controllers

#### Structure segmentation at controllers

In order to avoid jumps at the manipulated value (controller output) during setpoint changes, proportional and derivative actions can be switched into the feedback path. I.e.: The proportional action (proportionally) and the derivative action are then only influenced by the process value.




#### Switching proportional action into the feedback path

A proportion of the P action can be placed in the feedback using the `PropFacSP` parameter. Setpoint jumps then only affect the proportional action proportionally.

`PropFacSP = 0`: Proportional action is completely in the feedback path. Setpoint jumps do not affect the proportional action.

`PropFacSP = 1`: Proportional action is not in the feedback path. Proportional action is affected by setpoint and process variable (default setting).

#### Switching derivative action into the feedback path

The parameter `DiffToFbk` can be used to switch the D action into the feedback path. Setpoint jumps then no longer affect the derivative action directly.

`DiffToFbk = 0`: Derivative action is not in the feedback path. Derivative action is affected by setpoint and process variable (default setting).

`DiffToFbk = 1`: Derivative action is in the feedback path. Setpoint jumps do not affect the derivative action.

### Implementation of the structure segmentation

The implementation of the structure segmentation differs from the figure shown above such that changing `PropFacSP` does not change `MV`. Control behavior corresponds to the figure.

## 2.1.11 Messaging

### 2.1.11.1 Area of application of the alarm delays

### Area of application

A sensible area of application for setting alarm delays can, for example, be a motor. When it is started, an elevated starting current can occur and this could be reported depending on the configured limit. Since this usually settles down to a value below the set limit, the alarm would not make sense. In this case, the alarm delay that is intended to bridge the duration of the active alarm is used.

**Note**

Alarms that are really wanted are, naturally, also delayed when an alarm delay is used. Therefore select the delay period prudently!

### Alarm delays in the Advanced Process Library

There are three block types with a different application of the alarm delay for:

- One time value for all limits (Page 226)
- One time value per limit pair (Page 226)
- Two time values per limit pair (Page 227)
- Two time values for each individual limit (Page 228)

**Note**

An alarm delay is displayed with the following symbol in the limit view and in the faceplate overview:



### 2.1.11.2 One time value for all limits

#### Blocks with one time value for the alarm delay

This form of alarm delay is used for blocks without a unit designation in the name, for example, ConPerMon.

The alarm delay is used when brief violations of the set alarm thresholds are to be suppressed. The duration of the time delay is parameterized at the input `AlmDelay`. Parameter assignment is always carried out in seconds.

#### Activating the alarm delay

The alarm delay is disabled by default (`AlmDelay = 0`). To use the function, set a delay time [s] with the `AlmDelay` parameter.

### 2.1.11.3 One time value per limit pair

#### Blocks with one time value for the alarm delay per limit pair

The alarm delay is used when brief violations of the set alarm thresholds are to be suppressed.

**Note**

The message classes Alarm, Warning and Tolerance are not valid for user-configured message classes. Please take into consideration the validity of terms for user-configured message classes (Page 45).

The alarm delay is configured at the following inputs:

| Parameter for the delay time | Explanation |
|---|---|
| `XXX_A_DC` | Delay time for incoming events of the class<br>• Alarm<br>**XXX_A**larm_**D**elay**C**oming<br>XXX: Value to be monitored |
| `XXX_W_DC` | Delay time for incoming events of the class<br>• Warning<br>**XXX_W**arning_**D**elay**C**oming<br>XXX: Value to be monitored |

#### Activating the alarm delay

By default, the alarm delay is disabled for each limit, meaning that each individual parameter has 0 [s] pre-assigned.

To use the function, set a delay time [s] for each parameter.

### Pending alarms

Pending alarms, warnings, or tolerances are output at the corresponding output parameters:

- `XXX_AH_Act = 1`: Alarm limit (high) reached or violated
- `XXX_AL_Act = 1`: Alarm limit (low) reached or violated
- `XXX_WH_Act = 1`: Warning limit (high) reached or violated
- `XXX_WL_Act = 1`: Warning limit (low) reached or violated

If a message is active at one of these outputs, this is indicated by a 1.

### See also

User-configured message classes (Page 45)

## 2.1.11.4 Two time values per limit pair

### Blocks with two time values for the alarm delay per limit pair

The alarm delay is used when brief violations of the set alarm thresholds are to be suppressed.

**Note**

The message classes Alarm, Warning and Tolerance are not valid for user-configured message classes. Please take into consideration the validity of terms for user-configured message classes (Page 45).

The alarm delay is configured at the following inputs:

| Parameter for the delay time | Explanation |
|---|---|
| `XXX_A_DC` | Delay time for incoming events of the class<br>• Alarm<br>**XXX_A**larm_**D**elay**C**oming<br>XXX: Value to be monitored |
| `XXX_A_DG` | Delay time for outgoing events of the class<br>• Alarm<br>**XXX_A**larm_**D**elay**G**oing<br>XXX: Value to be monitored |
| `XXX_W_DC` | Delay time for incoming events of the class<br>• Warning<br>**XXX_W**arning_**D**elay**C**oming<br>XXX: Value to be monitored |
| `XXX_W_DG` | Delay time for outgoing events of the class<br>• Warning<br>**XXX_W**arning_**D**elay**G**oing<br>XXX: Value to be monitored |

| Parameter for the delay time | Explanation |
|---|---|
| XXX_T_DC | Delay time for incoming events of the class<br>• Tolerance<br>**XXX_T**olerance_**D**elay**C**oming<br>XXX: Value to be monitored |
| XXX_T_DG | Delay time for outgoing events of the class<br>• Tolerance<br>**XXX_T**olerance_**D**elay**G**oing<br>XXX: Value to be monitored |

### Activating the alarm delay

By default, the alarm delay is disabled for each individual pair, meaning that each individual parameter has 0 [s] pre-assigned.

To use the function, set a delay time [s] for each parameter.

### Pending alarms

Pending alarms, warnings, or tolerances are output at the corresponding output parameters:

- `XXX_AH_Act = 1`: Alarm limit (high) reached or violated
- `XXX_AL_Act = 1`: Alarm limit (low) reached or violated
- `XXX_WH_Act = 1`: Warning limit (high) reached or violated
- `XXX_WL_Act = 1`: Warning limit (low) reached or violated
- `XXX_TH_Act = 1`: Tolerance limit (high) reached or violated
- `XXX_TL_Act = 1`: Tolerance limit (low) reached or violated

If a message is active at one of these outputs, this is indicated by a 1.

### See also

User-configured message classes (Page 45)

## 2.1.11.5 Two time values for each individual limit

### Alarm delay for blocks with two time values for each individual limit

This form of alarm delay is used for the blocks PIDConL, PIDConR, and MonAnL.

**Note**

The message classes Alarm, Warning and Tolerance are not valid for user-configured message classes. Please take into consideration the validity of terms for user-configured message classes (Page 45).

The alarm delay is used when brief violations of the set alarm thresholds are to be suppressed. The alarm delay is configured at the following inputs:

| Parameter for the delay time | Explanation |
|---|---|
| XX_AH_DC | Delay time for incoming events of the class<br>• Alarm (for high limit)<br>**XX_A**larm**H**igh_**D**elay**C**oming<br>XX: Value to be monitored |
| XX_AH_DG | Delay time for outgoing events of the class<br>• Alarm (for high limit)<br>**XX_A**larm**H**igh_**D**elay**G**oing<br>XX: Value to be monitored |
| XX_AL_DC[1)] | Delay time for incoming events of the class<br>• Alarm (for low limit)<br>**XX_A**larm**L**ow_**D**elay**C**oming<br>XX: Value to be monitored |
| XX_AL_DG[1)] | Delay time for outgoing events of the class<br>• Alarm (for low limit)<br>**XX_A**larm**L**ow_**D**elay**G**oing<br>XX: Value to be monitored |
| XX_WH_DC | Delay time for incoming events of the class<br>• Warning (for high limit)<br>**XX_W**arning**H**igh_**D**elay**C**oming<br>XX: Value to be monitored |
| XX_WH_DG | Delay time for outgoing events of the class<br>• Warning (for high limit)<br>**XX_W**arning**H**igh_**D**elay**G**oing<br>XX: Value to be monitored |
| XX_WL_DC[1)] | Delay time for incoming events of the class<br>• Warning (for low limit)<br>**XX_W**arning**L**ow_**D**elay**C**oming<br>XX: Value to be monitored |
| XX_WL_DG[1)] | Delay time for outgoing events of the class<br>• Warning (for low limit)<br>**XX_W**arning**L**ow_**D**elay**G**oing<br>XX: Value to be monitored |
| XX_TH_DC | Delay time for incoming events of the class<br>• Tolerance (for high limit)<br>**XX_T**olerance**H**igh_**D**elay**C**oming<br>XX: Value to be monitored |
| XX_TH_DG | Delay time for outgoing events of the class<br>• Tolerance (for high limit)<br>**XX_T**olerance**H**igh_**D**elay**G**oing<br>XX: Value to be monitored |

| Parameter for the delay time | Explanation |
|---|---|
| XX_TL_DC[1)] | Delay time for incoming events of the class<br>• Tolerance (for low limit)<br>**XX_T**olerance**L**ow_**D**elay**C**oming<br>XX: Value to be monitored |
| XX_TL_DG[1)] | Delay time for outgoing events of the class<br>• Tolerance (for low limit)<br>**XX_T**olerance**L**ow_**D**elay**G**oing<br>XX: Value to be monitored |

[1)] In the "MonAnL" block, the inputs for a low delay time do not have the separate "L" letters (e.g. XX_AL_DC: the input name is XX_A_DC).

### Activating the alarm delay

By default, the alarm delay is disabled for each individual limit, meaning that each individual parameter has 0 [s] pre-assigned.

To use the function, set a delay time [s] for each parameter.

### Pending alarms

Pending alarms, warnings, or tolerances are output at the corresponding output parameters:

- `XX_AL_Act = 1`: Alarm limit (low) reached or violated
- `XX_AH_Act = 1`: Alarm limit (high) reached or violated
- `XX_WL_Act = 1`: Warning limit (low) reached or violated
- `XX_WH_Act = 1`: Warning limit (high) reached or violated
- `XX_TL_Act = 1`: Tolerance limit (low) reached or violated
- `XX_TH_Act = 1`: Tolerance limit (high) reached or violated

If a message is active at one of these outputs, this is indicated by a 1.

### See also

User-configured message classes (Page 45)

Separate delay times for each alarm (Page 181)

## 2.1.11.6 Generating instance-specific messages

### Generating instance-specific messages

You can generate instance-specific messages for a binary signal of every block.

The number of interconnectable input parameters that can be used freely varies with relation to the blocks. The X in the parameter name designates the position.

You can specify the following messages for these instance-specific messages:

- Message class
- Priority of the message
- Message text
- Message auxiliary value
- Acknowledge behavior

Additional information is available in the descriptions of the message functionality of the individual blocks and in the PCS 7 Configuration Manual Operator Station under "How to configure the user-specific messages".

**See also**

Time stamp (Page 231)

### 2.1.11.7 Suppressing messages using the MsgLock parameter

The `MsgLock = 1` parameter is used to selectively suppress the following messages depending on `Feature.Bit25`:

- All messages at the block

or

- All messages at the block, except for process control messages (for example, `CSF`, motor protection, feedback error) and external messages.

Messages already queued received the "outgoing" status with `MsgLock = 1` .

You can find additional information on `Feature.Bit25` in the section Suppression of all messages (Page 185)

**Note**

With the block OpAnL, this function is independent of `Feature.Bit25`.

### 2.1.11.8 Time stamp

**Time stamp**

The time stamp is the assignment of time information to the status change of a binary process signal. The status change of the signal is signaled together with the time information.

Use the EventTS block to report time stamped signals.

For more information on time stamping and how to configure it, please refer to the "PCS 7 - High-Precision Time Stamping Function Manual".

### Areas of application

Areas of application of the time stamp are for example:

- Accurately-timed detection of problems in process-related equipment. The time stamp enables you to explicitly identify signals that indicate the cause of the failure of a process unit.
- Analysis of system-wide interrelationships
- Detection and reporting of the sequence of time-critical signal changes

### Forming the time information

The time information is generated by one of the following methods and is specified at the block by means of the input parameter `TimeStampOn`:

- `TimeStampOn = 0`: Use time stamp of the CPU (default)
- `TimeStampOn = 1`: Use time stamp of the I/O

### Time stamping in the EventTS block

Connect the binary output parameter of another block (e.g. Pcs7DiIn) with a message input Inx (x = 1 ... 8) of the EventTS block.

When the EventTS block recognizes a change in the signal state at this message input, it uses the current time of the CPU as the time stamp. Only the signal changes that are slower than the cycle time of the block can be detected.

### High-precision time stamp in the process I/O

You have configured the hardware of your system for high-precision time stamping as explained in the "PCS 7 - High-Precision Time Stamping Function Manual". The signal changes are recognized in the I/O devices and the time stamp assigned to them. This data is available at the output parameter `TS_Out` of the Pcs7DiIT block.

The high-precision time stamp is independent of the cycle time of the blocks. The actual time resolution for two different status changes depends on your plant configuration and the hardware you are using.

Interconnect the output parameter `TS_Out` of Pcs7DiIT with a message input `InTSx(x = 1 ... 8)` of the EventTS block.

### Error handling

The system block ImDrvTs recognizes when the time stamp function in the I/O devices is defective and forwards this information to the Pcs7DiIT block. This then forms the time stamp using the current CPU time and sets the signal status of `TS_Out` output parameter to "Bad, due to device". The EventTS block then uses the current time of the CPU as the time stamp. You can find additional information in the "PCS 7 - High-Precision Time Stamping Function Manual".

#### 2.1.11.9 Connection of the time-stamped messages from EventTs or Event16Ts

If the output parameter `EventTsOut` of the block EventTs or Event16Ts is connected to the input parameter `EventTsIn` or `EventTs2In` of another block, the time-stamped messages of the block EventTs or Event16Ts will be displayed at the connected block. The following functionalities or parameters of the block are transferred to the connected message blocks:

- Maintenance release status (`MS_Release`)
- Operating mode "Out of service" (depends on `Feature2.Bit11`)
- Message lock with the parameter `MsgLock` (depends on `Feature2.Bit11`)
- Batch parameters (`BatchEn`, `Occupied`, `BatchID`, `BatchName`, and `StepNo`)

With `Feature2.Bit11` Suppress MsgLock and "Out of service" mode for a connected message block (Page 202), you can suppress the change to the "Out of service" mode and the message lock over `MsgLock`.

### 2.1.12 Settings for operator control and monitoring

#### 2.1.12.1 Display and operator input area for process values and setpoints

**Display and operator input area for process values and setpoints**

You specify the upper and lower area limits in the faceplate using interconnectable input parameters for the following:

- Display areas (bar display)
- Operator input area (for example setpoint and manipulated value)
- Input area for the limit values
  - Up to 7 numbers (including a decimal separator and a minus sign) are possible in the faceplates

The interconnectable input parameter is a structured variable that contains two analog values. Refer to the descriptions of the individual blocks for the relevant input parameters.

**Using a structured variable for scaling**

There are three possibilities with which you can influence the contents of the structured variables, namely with:

- The corresponding channel function block, for example, the FbAnIn block
- A conversion block, for example, the StruScIn block
- Direct parameter assignment at the block input

Additional information on data types is available in the documentation of CFC and STEP 7.

### 2.1.12.2 Opening additional faceplates

#### Opening additional faceplates

You can open standard views of other faceplates from various faceplate views. Here, you have the following options:

- Two buttons that you can assign freely and that are used to call faceplates of other blocks.
- Two predefined buttons for calling faceplates with a fixed assignment to the controller blocks.
- Buttons predefined for interlock functions

**Note**

**"Small" blocks**

With "Small" blocks, you can call up only one faceplate from the standard view.

#### Freely assignable buttons

From the standard view and from the preview, you can use a button to open the standard view of a block that can be selected freely. In order to use this function, in the CFC you need to interconnect the `SelFp1` input parameter for the button in the standard view or `SelFp2` for the button in the preview to any given output parameter of the block whose faceplate is to be opened. This makes the buttons in the faceplates visible.

**Note**

You can only configure the button in the standard view (`SelFp1`) with interlock blocks. There are no buttons with the GainSched block.

#### Button label

You can change the button labels in the "OS additional text" attribute to "SelFp1"/"SelFp2".

**Note**

There is also an alternative solution for labeling the buttons:

- Open the process picture in WinCC GraphicsDesigner.
- Open the object properties of the block icon.
- Under Configurations, assign the desired text to the attribute UserButtonText1 or UserButtonText2.

The main difference between this solution and the first solution is that the labels are only specified when the faceplate above the block icon is opened. This solution is therefore outdated.

### Predefined buttons for controller blocks

You can open the standard view for the following blocks from a controller standard view or parameter view (for example PIDConL):

- ConPerMon (can be called from the standard view)
  - To do this, you need to interconnect the output parameter `CPI` of the ConPerMon block to the input parameter `CPI_In` of the controller block.
- GainSched (can be called from the parameter view)
  - To do this, you need to interconnect the output parameter `Link2Gain` of the GainSched block to the input parameter `Gain` of the controller block.

The labels of the buttons cannot be changed here.

### Predefined buttons for interlocks

You can open the following interlock blocks from the standard view of the technology blocks:

- Activation enable
- Interlock without reset (interlock)
- Interlock with reset (protection)

The buttons intended for this are visible when the relevant input parameter (`Permit`, `Intlock` or `Protect`) is interconnected to an interlock block.

You can open the standard view for the following faceplates from the standard view of the interlock blocks:

- The blocks interconnected to the input values.
- The block interconnected to the output value.

The buttons intended for this are visible when the input parameters (for example `In01`) or the `Out` output parameter of the interlock block is interconnected to a block that has a faceplate.

**Note**

Interconnection of the `Out` output parameter to multiple blocks is not permitted. The reason for this is that a direct relationship must be established between the button in the faceplate and the faceplate to be opened by it.

## 2.1.12.3 Labeling of buttons and text

### Labeling of buttons and text

You can change the text of buttons (such as start/stop) for each specific instance. For binary OpDiXX faceplates, you can also change the static text for the "x command" for each specific instance.

To do this, you need to specify how the buttons are labeled yourself using the attributes "Text0" and "Text1" in the object properties of the block. These texts are displayed in the standard view and preview of the faceplate.

| Block type | Command button | Block I/O for parameter assignment "Text 1" |
|---|---|---|
| **DoseL** | Permission | `Permit.Value` |
| | Protection | `Protect.Value` |
| | Interlock | `Intlock.Value` |
| **FmCont** | Open | `OpenOp` |
| | Close | `CloseOp` |
| | Stop | `StopOp` |
| **FmTemp** | Open | `OpenOp` |
| | Close | `CloseOp` |
| | Stop | `StopOp` |
| **MonDiL** | Process value | `Out` |
| **MonDiS** | Process value | `Out` |
| **MotL** | Start | `StartMan` |
| | Stop | `StopMan` |
| | Rapid stop | `RapidStp` |
| | Permission | `Permit.Value` |
| | Protection | `Protect.Value` |
| | Permission stop | PermitSafePos.Value |
| | Interlock | `Intlock.Value` |
| **MotRevL** | Start - -> | `FwdMan` |
| | Start < - - | `RevMan` |
| | Stop | `StopMan` |
| | Rapid stop | `RapidStp` |
| | Permission | `Permit.Value` |
| | Protection | `Protect.Value` |
| | Interlock | `Intlock.Value` |
| | Permission fwd | `Permit.Value` |
| | Permission rev | `PermRev.Value` |
| | Permission stop | PermitSafePos.Value |
| | Protection fwd | `Protect.Value` |
| | Protection rev | `ProtRev.Value` |
| | Interlock fwd | `Intlock.Value` |
| | Interlock rev | `IntlRev.Value` |
| **MotS** | Start | `StartMan` |
| | Stop | `StopMan` |
| | Interlock | `Intlock.Value` |

| Block type | Command button | Block I/O for parameter assignment "Text 1" |
|---|---|---|
| **MotSpdCL** | Start \|-> | FwdMan |
| | Start < -\| | RevMan |
| | Stop | StopMan |
| | Rapid stop | RapidStp |
| | Permission | Permit.Value |
| | Permission stop | PermitSafePos.Value |
| | Protection | Protect.Value |
| | Interlock | Intlock.Value |
| **MotSpdL** | Start > | Spd1Man |
| | Start >> | Spd2Man |
| | Stop | StopMan |
| | Rapid stop | RapidStp |
| | Permission | Permit.Value |
| | Protection | Protect.Value |
| | Interlock | Intlock.Value |
| | Permission fast | Permit.Value |
| | Permission slow | PermSp2.Value |
| | Permission stop | PermitSafePos.Value |
| | Protection fast | Protect.Value |
| | Protection slow | ProtSp2.Value |
| | Interlock fast | Intlock.Value |
| | Interlock slow | IntlSp2.Value |
| **OpDi01** | Interlock | Intlock.Value |
| **OpDi03** | Interlock | Intlock.Value |
| **PIDConL** | Automatic | AutModOp |
| | Manual | ManModOp |
| | Program | AdvCoOn |
| | Out of service | OosOp |
| | External | SP_ExtOp |
| | Internal | SP_IntOp |
| | Interlock | Intlock.Value |
| **PIDConR** | Automatic | AutModOp |
| | Manual | ManModOp |
| | Program | AdvCoOn |
| | Out of service | OosOp |
| | External | SP_ExtOp |
| | Internal | SP_IntOp |
| | Interlock | Intlock.Value |
| **PIDStepL** | Open | OpenOp |
| | Close | CloseOp |
| | Stop | StopOp |

| Block type | Command button | Block I/O for parameter assignment "Text 1" |
|---|---|---|
| **VlvAnl** | Open | `OpenOp` |
| | Close | `CloseOp` |
| | Permission | `Permit.Value` |
| | Protection | `Protect.Value` |
| | Interlock | `Intlock.Value` |
| | Permission Open | `Permit.Value` |
| | Permission Close | `PermCls.Value` |
| | Permit neutral pos. | PermitSafePos.Value |
| | Protection Open | `Protect.Value` |
| | Protection Close | `ProtCls.Value` |
| | Interlock Open | `Intlock.Value` |
| | Interlock Close | `IntlCls.Value` |
| **VlvL** | Open | `OpenMan` |
| | Close | `CloseMan` |
| | Permission | `Permit.Value` |
| | Permit neutral pos. | PermitSafePos.Value |
| | Protection | `Protect.Value` |
| | Interlock | `Intlock.Value` |
| **Vlv2WayL** | Pos0 | `Pos0Man` |
| | Pos1 | `Pos1Man` |
| | Pos2 | `Pos2Man` |
| | Permission | `Permit.Value` |
| | Protection | `Protect.Value` |
| | Interlock | `Intlock.Value` |
| **VlvMotL** | Open | `OpenMan` |
| | Close | `CloseMan` |
| | Stop | `StopMan` |
| | Rapid stop | `RapidStp` |
| | Permission | `Permit.Value` |
| | Protection | `Protect.Value` |
| | Interlock | `Intlock.Value` |
| | Permission Open | `Permit.Value` |
| | Permission Close | `PermCls.Value` |
| | Permit neutral pos. or Permission stop | PermitSafePos.Value |
| | Protection Open | `Protect.Value` |
| | Protection Close | `ProtCls.Value` |
| | Interlock Open | `Intlock.Value` |
| | Interlock Close | `IntlCls.Value` |

| Block type | Command button | Block I/O for parameter assignment "Text 1" |
|---|---|---|
| **VlvPosL** | Open | OpenMan |
| | Close | CloseMan |
| | Stop | StopMan |
| | Rapid stop | RapidStp |
| | Permission | Permit.Value |
| | Protection | Protect.Value |
| | Interlock | Intlock.Value |
| | Permission Open | Permit.Value |
| | Permission Close | PermCls.Value |
| | Permit neutral pos. or Permission stop | PermitSafePos.Value |
| | Protection Open | Protect.Value |
| | Protection Close | ProtCls.Value |
| | Interlock Open | Intlock.Value |
| | Interlock Close | IntlCls.Value |
| **VlvS** | Open | OpenMan |
| | Close | CloseMan |
| | Interlock | Intlock.Value |

The default text is shown if no text is configured.

If the text is longer than can be displayed with the default font size, the font size is automatically reduced until the text is fully shown. The smallest font size is 7 point.

Refer to the function description of the faceplate to learn which parameters are affected by this function.

### 2.1.12.4 Displaying auxiliary values

## Displaying auxiliary values

Up to two auxiliary values can be displayed in the standard view of some faceplates. This feature can be used, for example, with motors to indicate the motor current and winding temperature.

To do so interconnect the value that you want to have displayed with the input parameters UserAna1 or UserAna2.

In the object properties (I/Os > Identifier) of the block in CFC, you can specify the text to be displayed for these parameters in the standard view of the faceplate.

### 2.1.12.5 Selecting a unit of measure

#### Coded unit of measure

The parameter `XXX_Unit` is used to specify the unit of measure for the corresponding input parameter (XXX stands for a specific parameter, for example, `PV_Unit`). Entry is carried out in the form of a code. Exactly one unit of measure is assigned to each code and is displayed on the faceplate.

You can interconnect the `XXX_Unit` input parameter of a technologic block with the `XXXUnit` output parameter of an analog input channel block. At the analog input channel block, enter the unit of measure at the `XXXUnit` input parameter (`XXX` stands for a specific parameter, for example `PV_InUnit, PVOutUnit`).

If the parameter value of `XXX_Unit` is out of range (that is, the value is not defined), the faceplate displays "!undef.!" in the place of unit.

**Note**

**Special notes for channel blocks PCS7AnIn, PCS7AnOu, FbAnIn and FbAnOu**

You can use the S7_enum attribute to display the unit in plain text in the CFC editor for these blocks.

**Note**

In block icons, the update/refresh time of the units is 1 h. If the unit is changed from CFC, the new unit will be visible in the block icon after 1 hour or if the process picture is reloaded. In faceplates, the update/refresh time of the units is 5 seconds, so the new unit will be visible after 5 s.

#### Customer-specific units

It is possible to use units which differ from the IEC 61158-2 standard. You can define your own units in the range 1 to 199 in an XML file.

The name of the XML should be APLCustomerUnits.xml and should be placed in the project path in the folder "GraCS" on both server and client.

Below is an example to describe the content of the XML file APLCustomerUnits.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<UserDefinedUnits>                          <!-- Root node start -->
    <!-- Define the first unit for different languages -->
    <Unit id="1">                           <!-- first node with unit ID 1 start -->
      <Name lcid="1031">German1</Name>      <!-- unit value (e.g. German1) in German -->
      <Name lcid="1033">English1</Name>     <!-- unit value (e.g. English1) in English -->
      <Name lcid="1034">Spanish1</Name>     <!-- unit value (e.g. Spanish1) in Spanish -->
      <Name lcid="1036">French1</Name>      <!-- unit value (e.g. French1) in French -->
      <Name lcid="1040">Italian1</Name>     <!-- unit value (e.g. Italian1) in Italian -->
      <Name lcid="1041">Japanese1</Name>    <!-- unit value (e.g. Japanese1) in Japanese -->
      <Name lcid="2052">Chinese1</Name>     <!-- unit value (e.g. Chinese1) in Chinese -->
```

```
    </Unit>                              <!-- first node with unit ID 1 end -->
    <!-- Define the second unit for different languages -->
    <Unit id="2">
      <Name lcid="1031">German2</Name>     <!-- lcid="1031" is for German -->
      <Name lcid="1033">English2</Name>    <!-- lcid="1033" is for English -->
      <Name lcid="1034">Spanish2</Name>    <!-- lcid="1034" is for Spanish -->
      <Name lcid="1036">French2</Name>     <!-- lcid="1036" is for French -->
      <Name lcid="1040">Italian2</Name>    <!-- lcid="1040" is for Italian -->
      <Name lcid="1041">Japanese2</Name>   <!-- lcid="1041" is for Japanese -->
      <Name lcid="2052">Chinese2</Name>    <!-- lcid="2052" is for Chinese -->
    </Unit>
<!-- similarly define other units up to unit id = 199-->
</UserDefinedUnits>                      <!-- Root node end -->
```

The "Unit id" value should be given to `XXX_Unit` parameter in the function blocks.

**Note**

The XML is read in only once when WinCC runtime is activated. If any changes are done in the XML later, you must deactivate, close, and activate the WinCC runtime again to reflect the changes.

### Using the unit of measure with controllers for the ConPerMon block

For controller blocks, the current unit of measure is output via output parameter `XX_UnitOut` . If you use the ConPerMon block, you must switch this output parameter with the corresponding input parameter `XXX_Unit` on the ConPerMon block.

### Using the S7_unit attribute

If you set the `xxx_Unit` parameter to 0, the entry is displayed by the `S7_Unit` attribute in the faceplate and in the block symbol.

### Overview of the units of measure

The units of measure are listed in the following tables:

List of the most commonly used units of measure in accordance with specification PROFIBUS PA Profile for Process Control Devices up to version PNO V3.02.

**Note**

Due to space constraints on the system, not all units referred to can be displayed in totality. The display of the measurement units is left-aligned. If you want to abbreviate the unit of measurement, use the function **Customer-specific units**.

| Value | Display | Description |
|---|---|---|
| 1000 | K | Kelvin |
| 1001 | °C | Degrees Celsius |
| 1002 | °F | Degrees Fahrenheit |
| 1005 | ° | Degree |
| 1006 | ' | Minute |
| 1007 | " | Second |
| 1010 | m | Meter |
| 1013 | mm | Millimeter |
| 1018 | ft | Foot |
| 1023 | $m^2$ | Square meter |
| 1038 | L | Liter |
| 1041 | hl | Hectoliter |
| 1054 | s | Second |
| 1058 | min | Minute |
| 1059 | h | Hour |
| 1060 | d | Day |
| 1061 | m/s | Meters per second |
| 1077 | Hz | Hertz |
| 1081 | kHz | Kilohertz |
| 1082 | 1/s | Per second |
| 1083 | 1/min | Per minute |
| 1088 | kg | Kilogram |
| 1092 | t | Metric ton |
| 1100 | $g/cm^3$ | Grams per cubic centimeter |
| 1105 | g/L | Grams per liter |
| 1120 | N | Newton |
| 1123 | mN | Millinewton |
| 1130 | Pa | Pascal |
| 1133 | kPa | Kilopascal |
| 1137 | bar | Bar |
| 1138 | mbar | Millibar |
| 1149 | $mmH_2O$ | Millimeters of water[1] |
| 1175 | W·h | Watt hour |
| 1179 | kW·h | Kilowatt hour |
| 1181 | $kcal_{th}$ | Kilocalories[1] |
| 1190 | kW | Kilowatt |
| 1209 | A | Ampere |
| 1211 | mA | Milliampere |
| 1221 | A·h | Ampere hour |
| 1240 | V | Volt |
| 1349 | $m^3/h$ | Cubic meters per hour |
| 1353 | L/h | Liters per hour |

| Value | Display | Description |
|---|---|---|
| 1384 | mol | Mol |
| 1422 | pH | pH value |

List of all units of measure in accordance with specification PROFIBUS PA Profile for Process Control Devices up to version PNO V3.02.

| Value | Display | Description |
|---|---|---|
| 1000 | K | Kelvin |
| 1001 | °C | Degrees Celsius |
| 1002 | °F | Degrees Fahrenheit |
| 1003 | °R | Degree Rankine |
| 1004 | rad | Radian |
| 1005 | ° | Degree |
| 1006 | ' | Minute |
| 1007 | " | Second |
| 1008 | gon | Gon |
| 1009 | r | Revolution |
| 1010 | m | Meter |
| 1011 | km | Kilometer |
| 1012 | cm | Centimeter |
| 1013 | mm | Millimeter |
| 1014 | µm | Micrometer |
| 1015 | nm | Nanometer |
| 1016 | pm | Picometer |
| 1017 | Å | Angstrom ² |
| 1018 | ft | Foot |
| 1019 | in | Inch |
| 1020 | yd | Yard |
| 1021 | mile | Mile |
| 1022 | nautical mile | Nautical mile |
| 1023 | $m^2$ | Square meter |
| 1024 | $km^2$ | Square kilometer |
| 1025 | $cm^2$ | Square centimeter |
| 1026 | $dm^2$ | Square decimeter |
| 1027 | $mm^2$ | Square millimeter |
| 1028 | a | Are |
| 1029 | ha | Hectare |
| 1030 | $in^2$ | Square inch |
| 1031 | $ft^2$ | Square foot |
| 1032 | $yd^2$ | Square yard |
| 1033 | $mile^2$ | Square mile |
| 1034 | $m^3$ | Cubic meter |
| 1035 | $dm^3$ | Cubic decimeter |
| 1036 | $cm^3$ | Cubic centimeter |

| Value | Display | Description |
|---|---|---|
| 1037 | $mm^3$ | Cubic millimeter |
| 1038 | L | Liter |
| 1039 | cl | Centiliter |
| 1040 | ml | Milliliter |
| 1041 | hl | Hectoliter |
| 1042 | $in^3$ | Cubic inch |
| 1043 | $ft^3$ | Cubic foot |
| 1044 | $yd^3$ | Cubic yard |
| 1045 | $mile^3$ | Cubic mile |
| 1046 | pint | Pint |
| 1047 | quart | Quart |
| 1048 | gal | US gallon |
| 1049 | ImpGal | Imperial gallon |
| 1050 | bushel | Bushel |
| 1051 | bbl | Barrel = 42 gallons |
| 1052 | bbl(liq) | Liquid barrel = 31.5 gallons |
| 1053 | $ft^3$ std. | Standard cubic foot |
| 1054 | s | Second |
| 1055 | ks | Kilosecond |
| 1056 | ms | Millisecond |
| 1057 | µs | Microsecond |
| 1058 | min | Minute |
| 1059 | h | Hour |
| 1060 | d | Day |
| 1061 | m/s | Meters per second |
| 1062 | mm/s | Millimeters per second |
| 1063 | m/h | Meters per hour |
| 1064 | km/h | Kilometers per hour |
| 1065 | knot | Knot |
| 1066 | in/s | Inches per second |
| 1067 | ft/s | Feet per second |
| 1068 | yd/s | Yards per second |
| 1069 | in/min | Inches per minute |
| 1070 | ft/min | Feet per minute |
| 1071 | yd/min | Yards per minute |
| 1072 | in/h | Inches per hour |
| 1073 | ft/h | Feet per hour |
| 1074 | yd/h | Yards per hour |
| 1075 | mi/h | Miles per hour |
| 1076 | $m/s^2$ | Meter/second squared |
| 1077 | Hz | Hertz |
| 1078 | THz | Terahertz |
| 1079 | GHz | Gigahertz |

| Value | Display | Description |
|---|---|---|
| 1080 | MHz | Megahertz |
| 1081 | kHz | Kilohertz |
| 1082 | 1/s | Per second |
| 1083 | 1/min | Per minute |
| 1084 | r/s | Revolutions per second |
| 1085 | rpm | Revolutions per minute |
| 1086 | rad/s | Radians per second |
| 1087 | $1/s^2$ | Per second squared |
| 1088 | kg | Kilogram |
| 1089 | g | Gram |
| 1090 | mg | Milligram |
| 1091 | Mg | Megagram |
| 1092 | t | Metric ton |
| 1093 | oz | Ounce |
| 1094 | lb | Pound |
| 1095 | STon | US ton (short ton) |
| 1096 | LTon | British ton (long ton) |
| 1097 | $kg/m^3$ | Kilograms per cubic meter |
| 1098 | $Mg/m^3$ | Megagrams per cubic meter |
| 1099 | $kg/dm^3$ | Kilograms per cubic decimeter |
| 1100 | $g/cm^3$ | Grams per cubic centimeter |
| 1101 | $g/m^3$ | Grams per cubic meter |
| 1102 | $t/m^3$ | Metric tons per cubic meter |
| 1103 | kg/L | Kilogram per liter |
| 1104 | g/ml | Grams per milliliter |
| 1105 | g/L | Grams per liter |
| 1106 | $lb/in^3$ | Pounds per cubic inch |
| 1107 | $lb/ft^3$ | Pounds per cubic foot |
| 1108 | lb/gal | Pounds per US gallon |
| 1109 | $STon/yd^3$ | US tons per cubic yard |
| 1110 | °Twad | Degree Twaddell |
| 1111 | °Baum (hv) | Degree Baumé (heavy) |
| 1112 | °Baum (lt) | Degree Baumé (light) |
| 1113 | °API | Degrees API |
| 1114 | SGU | Specific gravity units |
| 1115 | kg/m | Kilograms per meter |
| 1116 | mg/m | Milligrams per meter |
| 1117 | tex | Tex |
| 1118 | $kg \cdot m^2$ | Kilograms per square meter |
| 1119 | kg·m/s | Kilograms per meter per second |
| 1120 | N | Newton |
| 1121 | MN | Meganewton |
| 1122 | kN | Kilonewton |

| Value | Display | Description |
|---|---|---|
| 1123 | mN | Millinewton |
| 1124 | µN | Micronewton |
| 1125 | $kg \cdot m^2/s$ | Kilograms per square meter per second |
| 1126 | N·m | Newton meter |
| 1127 | MN·m | Meganewton meter |
| 1128 | kN·m | Kilonewton meter |
| 1129 | mN·m | Millinewton meter |
| 1130 | Pa | Pascal |
| 1131 | GPa | Gigapascal |
| 1132 | MPa | Megapascal |
| 1133 | kPa | Kilopascal |
| 1134 | mPa | Millipascal |
| 1135 | µPa | Micropascal |
| 1136 | hPa | Hectopascal |
| 1137 | bar | Bar |
| 1138 | mbar | Millibar |
| 1139 | torr | Torr |
| 1140 | atm | Atmosphere |
| 1141 | psi | Pounds per square inch |
| 1142 | psia | Pounds per square inch (absolute) |
| 1143 | psig | Pounds per square inch (gauge) |
| 1144 | $g/cm^2$ | Grams per square centimeter |
| 1145 | $kg/cm^2$ | Kilograms per square centimeter |
| 1146 | $inH_2O$ | Inches of water |
| 1147 | $inH_2O$ (4°C) | Inches of water at 4 degrees Celsius |
| 1148 | $inH_2O$ (68°F) | Inches of water at 68 degrees Fahrenheit |
| 1149 | $mmH_2O$ | Millimeters of water[1] |
| 1150 | $mmH_2O$ (4°C) | Millimeters of water at 4 degrees Celsius[1] |
| 1151 | $mmH_2O$ (68°F) | Millimeters of water at 68 degrees Fahrenheit[1] |
| 1152 | $ftH_2O$ | Feet of water[1] |
| 1153 | $ftH_2O$ (4°C) | Feet of water at 4 degrees Celsius[1] |
| 1154 | $ftH_2O$ (68°F) | Feet of water at 68 degrees Fahrenheit[1] |
| 1155 | inHg | Inches of mercury |
| 1156 | inHg (0°C) | Inches of mercury at 0 degrees Celsius |
| 1157 | mmHg | Millimeters of mercury |
| 1158 | mmHg (0°C) | Millimeters of mercury at 0 degrees Celsius |
| 1159 | Pa·s | Pascal second |

| Value | Display | Description |
|---|---|---|
| 1160 | $m^2/s$ | Square meters per second |
| 1161 | P | Poise |
| 1162 | cP | Centipoise |
| 1163 | St | Stokes |
| 1164 | cSt | Centistokes |
| 1165 | N/m | Newtons per meter |
| 1166 | mN/m | Millinewtons per meter |
| 1167 | J | Joule |
| 1168 | EJ | Exajoule |
| 1169 | PJ | Petajoule |
| 1170 | TJ | Terajoule |
| 1171 | GJ | Gigajoule |
| 1172 | MJ | Megajoule |
| 1173 | kJ | Kilojoule |
| 1174 | mJ | Millijoule |
| 1175 | W·h | Watt hour |
| 1176 | TW·h | Terawatt hour |
| 1177 | GW·h | Gigawatt hour |
| 1178 | MW·h | Megawatt hour |
| 1179 | kW·h | Kilowatt hour |
| 1180 | $cal_{th}$ | Calorie (thermo chemical)[1] |
| 1181 | $kcal_{th}$ | Kilocalorie (thermo chemical)[1] |
| 1182 | $Mcal_{th}$ | Megacalorie (thermo chemical)[1] |
| 1183 | $Btu_{th}$ | British thermal unit[1] |
| 1184 | datherm | Decatherm |
| 1185 | ft·lbf | Foot pound |
| 1186 | W | Watt |
| 1187 | TW | Terawatt |
| 1188 | GW | Gigawatt |
| 1189 | MW | Megawatt |
| 1190 | kW | Kilowatt |
| 1191 | mW | Milliwatt |
| 1192 | µW | Microwatt |
| 1193 | nW | Nanowatt |
| 1194 | pW | Picowatt |
| 1195 | $Mcal_{th}/h$ | Megacalorie per hour[1] |
| 1196 | MJ/h | Megajoule per hour |
| 1197 | $Btu_{th}/h$ | British thermal units per hour[1] |
| 1198 | hp | Horsepower |
| 1199 | W/(m·K) | Watts per meter kelvin |
| 1200 | $W/(m^2 \cdot K)$ | Watts per (square meter kelvin) |
| 1201 | $m^2 \cdot K/W$ | Square meters kelvin per Watt |
| 1202 | J/K | Joules per kelvin |

| Value | Display | Description |
|---|---|---|
| 1203 | kJ/K | Kilojoules per kelvin |
| 1204 | J/(kg·K) | Joules per (kilogram kelvin) |
| 1205 | kJ/(kg·K) | Kilojoules per (kilogram kelvin) |
| 1206 | J/kg | Joules per kilogram |
| 1207 | MJ/kg | Megajoules per kilogram |
| 1208 | kJ/kg | Kilojoules per kilogramm |
| 1209 | A | Ampere |
| 1210 | kA | Kiloampere |
| 1211 | mA | Milliampere |
| 1212 | µA | Microampere |
| 1213 | nA | Nanoampere |
| 1214 | pA | Picoampere |
| 1215 | C | Coulomb |
| 1216 | MC | Megacoulomb |
| 1217 | kC | Kilocoulomb |
| 1218 | µC | Microcoulomb |
| 1219 | nC | Nanocoulomb |
| 1220 | pC | Picocoulomb |
| 1221 | A·h | Ampere hour |
| 1222 | $C/m^3$ | Coulombs per cubic meter |
| 1223 | $C/mm^3$ | Coulombs per cubic millimeter |
| 1224 | $C/cm^3$ | Coulombs per cubic centimeter |
| 1225 | $kC/m^3$ | Kilocoulombs per cubic meter |
| 1226 | $mC/m^3$ | Millicoulombs per cubic meter |
| 1227 | $µC/m^3$ | Microcoulombs per cubic meter |
| 1228 | $C/m^2$ | Coulombs per square meter |
| 1229 | $C/mm^2$ | Coulombs per square millimeter |
| 1230 | $C/cm^2$ | Coulombs per square centimeter |
| 1231 | $kC/m^2$ | Kilocoulombs per square meter |
| 1232 | $mC/m^2$ | Millicoulombs per square meter |
| 1233 | $µC/m^2$ | Microcoulombs per square meter |
| 1234 | V/m | Volts per meter |
| 1235 | MV/m | Megavolts per meter |
| 1236 | kV/m | Kilovolts per meter |
| 1237 | V/cm | Volts per centimeter |
| 1238 | mV/m | Millivolts per meter |
| 1239 | µV/m | Microvolts per meter |
| 1240 | V | Volt |
| 1241 | MV | Megavolt |
| 1242 | kV | Kilovolt |
| 1243 | mV | Millivolt |
| 1244 | µV | Microvolt |
| 1245 | F | Farad |

| Value | Display | Description |
|---|---|---|
| 1246 | mF | Millifarad |
| 1247 | µF | Microfarad |
| 1248 | nF | Nanofarad |
| 1249 | pF | Picofarad |
| 1250 | F/m | Farad per meter |
| 1251 | µF/m | Microfarad per meter |
| 1252 | nF/m | Nanofarad per meter |
| 1253 | pF/m | Picofarad per meter |
| 1254 | C·m | Coulomb meter |
| 1255 | $A/m^2$ | Amperes per square meter |
| 1256 | $MA/m^2$ | Megaamperes per square meter |
| 1257 | $A/cm^2$ | Amperes per square centimeter |
| 1258 | $kA/m^2$ | Kiloamperes per square meter |
| 1259 | A/m | Amperes per meter |
| 1260 | kA/m | Kiloamperes per meter |
| 1261 | A/cm | Amperes per centimeter |
| 1262 | T | Tesla |
| 1263 | mT | Millitesla |
| 1264 | µT | Microtesla |
| 1265 | nT | Nanotesla |
| 1266 | Wb | Weber |
| 1267 | mWb | Milliweber |
| 1268 | Wb/m | Webers per meter |
| 1269 | kWb/m | Kilowebers per meter |
| 1270 | H | Henry |
| 1271 | mH | Millihenry |
| 1272 | µH | Microhenry |
| 1273 | nH | Nanohenry |
| 1274 | pH | Picohenry |
| 1275 | H/m | Henries per meter |
| 1276 | µH/m | Microhenries per meter |
| 1277 | nH/m | Nanohenries per meter |
| 1278 | $A·m^2$ | Ampere square meters |
| 1279 | $N·m^2/A$ | Newton meter squared per ampere |
| 1280 | Wb·m | Weber meter |
| 1281 | Ω | Ohm[1] |
| 1282 | GΩ | Gigaohm[1] |
| 1283 | MΩ | Megaohm[1] |
| 1284 | kΩ | Kiloohm[1] |
| 1285 | mΩ | Milliohm[1] |
| 1286 | µΩ | Microohm[1] |
| 1287 | S | Siemens |
| 1288 | kS | Kilosiemens |

| Value | Display | Description |
|---|---|---|
| 1289 | mS | Millisiemens |
| 1290 | µS | Microsiemens |
| 1291 | Ω·m | Ohms times meters[1] |
| 1292 | GΩ·m | Gigaohms times meters[1] |
| 1293 | MΩ·m | Megaohms times meters[1] |
| 1294 | kΩ·m | Kiloohms times meters[1] |
| 1295 | Ω·cm | Ohms times centimeters[1] |
| 1296 | mΩ·m | Milliohms times meters[1] |
| 1297 | µΩ·m | Microohms times meters[1] |
| 1298 | nΩ·m | Nanoohms times meters[1] |
| 1299 | S/m | Siemens per meter |
| 1300 | MS/m | Megasiemens per meter |
| 1301 | kS/m | Kilosiemens per meter |
| 1302 | mS/cm | Millisiemens per centimeter |
| 1303 | µS/mm | Microsiemens per millimeter |
| 1304 | 1/H | Per henry |
| 1305 | sr | Steradian |
| 1306 | W/sr | Watts per steradian |
| 1307 | W/(sr·m$^2$) | Watts per (steradian square meter) |
| 1308 | W/(m$^2$) | Watts per square meter |
| 1309 | lm | Lumen |
| 1310 | lm·s | Lumen second |
| 1311 | lm·h | Lumen hour |
| 1312 | lm/m$^2$ | Lumens per square meter |
| 1313 | lm/W | Lumens per watt |
| 1314 | lx | Lux |
| 1315 | lx·s | Lux second |
| 1316 | cd | Candela |
| 1317 | cd/m$^2$ | Candela per square meter |
| 1318 | g/s | Grams per second |
| 1319 | g/min | Grams per minute |
| 1320 | g/h | Grams per hour |
| 1321 | g/d | Grams per day |
| 1322 | kg/s | Kilograms per second |
| 1323 | kg/min | Kilograms per minute |
| 1324 | kg/h | Kilograms per hour |
| 1325 | kg/d | Kilograms per day |
| 1326 | t/s | Metric tons per second |
| 1327 | t/min | Metric tons per minute |
| 1328 | t/h | Metric tons per hour |
| 1329 | t/d | Metric tons per day |
| 1330 | lb/s | Pounds per second |
| 1331 | lb/min | Pounds per minute |

| Value | Display | Description |
|---|---|---|
| 1332 | lb/h | Pounds per hour |
| 1333 | lb/d | Pounds per day |
| 1334 | STon/s | US tons per second |
| 1335 | STon/min | US tons per minute |
| 1336 | STon/h | US tons per hour |
| 1337 | STon/d | US tons per day |
| 1338 | LTon/s | British tons per second |
| 1339 | LTon/min | British tons per minute |
| 1340 | LTon/h | British tons per hour |
| 1341 | LTon/d | British tons per day |
| 1342 | % | Percent |
| 1343 | % sol/wt | Percentage solids per weight unit |
| 1344 | % sol/vol | Percentage solids per volume unit |
| 1345 | % stm qual | Percentage steam quality |
| 1346 | °Plato | Degree plato |
| 1347 | $m^3/s$ | Cubic meters per second |
| 1348 | $m^3/min$ | Cubic meters per minute |
| 1349 | $m^3/h$ | Cubic meters per hour |
| 1350 | $m^3/d$ | Cubic meters per day |
| 1351 | L/s | Liters per second |
| 1352 | L/min | Liters per minute |
| 1353 | L/h | Liters per hour |
| 1354 | L/d | Liters per day |
| 1355 | ML/d | Megaliters per day |
| 1356 | $ft^3/s$ | Cubic feet per second |
| 1357 | $ft^3/m$ | Cubic feet per minute |
| 1358 | $ft^3/h$ | Cubic feet per hour |
| 1359 | $ft^3/d$ | Cubic feet per day |
| 1360 | $ft^3/min$ std | Standard cubic feet per minute |
| 1361 | $ft^3/h$ std | Standard cubic feet per hour |
| 1362 | gal/s | US gallons per second |
| 1363 | gal/min | US gallons per minute |
| 1364 | gal/h | US gallons per hour |
| 1365 | gal/d | US gallons per day |
| 1366 | Mgal/d | Mega US gallons per day |
| 1367 | ImpGal/s | Imperial gallons per second |
| 1368 | ImpGal/min | Imperial gallons per minute |
| 1369 | ImpGal/h | Imperial gallons per hour |
| 1370 | ImpGal/d | Imperial gallons per day |
| 1371 | bbl/s | Barrels per second |
| 1372 | bbl/min | Barrels per minute |
| 1373 | bbl/h | Barrels per hour |
| 1374 | bbl/d | Barrels per day |

| Value | Display | Description |
|---|---|---|
| 1375 | $W/m^2$ | Watts per square meter |
| 1376 | $mW/m^2$ | Milliwatts per square meter |
| 1377 | $\mu W/m^2$ | Microwatts per square meter |
| 1378 | $pW/m^2$ | Picowatts per square meter |
| 1379 | $Pa \cdot s/m^3$ | Pascal seconds per cubic meter |
| 1380 | N·s/m | Newton seconds per meter |
| 1381 | Pa·s/m | Pascal seconds per meter |
| 1382 | B | Bel |
| 1383 | dB | Decibel |
| 1384 | mol | Mol |
| 1385 | kmol | Kilomole |
| 1386 | mmol | Millimole |
| 1387 | µmol | Micromole |
| 1388 | kg/mol | Kilograms per mole |
| 1389 | g/mol | Grams per mole |
| 1390 | $m^3/mol$ | Cubic meters per mole |
| 1391 | $dm^3/mol$ | Cubic decimeters per mole |
| 1392 | $cm^3/mol$ | Cubic centimeters per mole |
| 1393 | L/mol | Liters per mole |
| 1394 | J/mol | Joules per mole |
| 1395 | kJ/mol | Kilojoules per mole |
| 1396 | J/(mol·K) | Joules per mole kelvin |
| 1397 | $mol/m^3$ | Moles per cubic meter |
| 1398 | $mol/dm^3$ | Moles per cubic decimeter |
| 1399 | mol/L | Moles per liter |
| 1400 | mol/kg | Moles per kilogram |
| 1401 | mmol/kg | Millimoles per kilogram |
| 1402 | Bq | Becquerel |
| 1403 | MBq | Megabecquerel |
| 1404 | kBq | Kilobecquerel |
| 1405 | Bq/kg | Becquerels per kilogram |
| 1406 | kBq/kg | Kilobecquerels per kilogram |
| 1407 | MBq/kg | Megabecquerels per kilogram |
| 1408 | Gy | Gray |
| 1409 | mGy | Milligray |
| 1410 | rd | Rad |
| 1411 | Sv | Sievert |
| 1412 | mSv | Millisievert |
| 1413 | rem | Rem |
| 1414 | C/kg | Coulombs per kilogram |
| 1415 | mC/kg | Millicoulombs per kilogram |
| 1416 | R | Röntgen |
| 1417 | $1/Jm^3$ | Density of magnetic energy |

| Value | Display | Description |
|---|---|---|
| 1418 | $e/Vm^3$ | |
| 1419 | $m^3/C$ | Cubic meters per coulomb |
| 1420 | V/K | Volts per kelvin |
| 1421 | mV/K | Millivolts per kelvin |
| 1422 | pH | pH value |
| 1423 | ppm | Parts per million |
| 1424 | ppb | Parts per billion |
| 1425 | ppth | Parts per trillion |
| 1426 | °Brix | Degrees Brix |
| 1427 | °Ball | Degrees Balling |
| 1428 | proof/vol | Proof per volume |
| 1429 | proof/mass | Proof per mass |
| 1430 | lb/ImpGal | Pounds per Imperial gallon |
| 1431 | $kcal_{th}/s$ | Kilocalories per second[1] |
| 1432 | $kcal_{th}/min$ | Kilocalories per minute[1] |
| 1433 | $kcal_{th}/h$ | Kilocalories per hour[1] |
| 1434 | $kcal_{th}/d$ | Kilocalories per day[1] |
| 1435 | $Mcal_{th}/s$ | Megacalories per second[1] |
| 1436 | $Mcal_{th}/min$ | Megacalories per minute[1] |
| 1437 | $Mcal_{th}/d$ | Megacalories per day[1] |
| 1438 | kJ/s | Kilojoules per second |
| 1439 | kJ/min | Kilojoules per minute |
| 1440 | kJ/h | Kilojoule per hour |
| 1441 | kJ/d | Kilojoules per day |
| 1442 | MJ/s | Megajoules per second |
| 1443 | MJ/min | Megajoules per minute |
| 1444 | MJ/d | Megajoules per day |
| 1445 | $Btu_{th}/s$ | British thermal units per second[1] |
| 1446 | $Btu_{th}/min$ | British thermal units per minute[1] |
| 1447 | $Btu_{th}/d$ | British thermal units per day[1] |
| 1448 | µgal/s | Micro US gallons per second |
| 1449 | mgal/s | Milli US gallons per second |
| 1450 | kgal/s | Kilo US gallons per second |
| 1451 | Mgal/s | Mega US gallons per second |
| 1452 | µgal/min | Micro US gallons per minute |
| 1453 | mgal/min | Milli US gallons per minute |
| 1454 | kgal/min | Kilo US gallons per minute |
| 1455 | Mgal/min | Mega US gallons per minute |
| 1456 | µgal/h | Micro US gallons per hour |
| 1457 | mgal/h | Milli US gallons per hour |
| 1458 | kgal/h | Kilo US gallons per hour |
| 1459 | Mgal/h | Mega US gallons per hour |
| 1460 | µgal/d | Micro US gallons per day |

| Value | Display | Description |
|---|---|---|
| 1461 | mgal/d | Milli US gallons per day |
| 1462 | kgal/d | Kilo US gallons per day |
| 1463 | µImpGal/s | Micro Imperial gallons per second |
| 1464 | mImpGal/s | Milli Imperial gallons per second |
| 1465 | kImpGal/s | Kilo Imperial gallons per second |
| 1466 | MImpGal/s | Mega Imperial gallons per second |
| 1467 | µImpGal/min | Micro Imperial gallons per minute |
| 1468 | mImpGal/min | Milli Imperial gallons per minute |
| 1469 | kImpGal/min | Kilo Imperial gallons per minute |
| 1470 | MImpGal/min | Mega Imperial gallons per minute |
| 1471 | µImpGal/h | Micro Imperial gallons per hour |
| 1472 | mImpGal/h | Milli Imperial gallons per hour |
| 1473 | kImpGal/h | Kilo Imperial gallons per hour |
| 1474 | MImpGal/h | Mega Imperial gallons per hour |
| 1475 | µImpgal/d | Micro Imperial gallons per day |
| 1476 | mImpgal/d | Milli Imperial gallons per day |
| 1477 | kImpgal/d | Kilo Imperial gallons per day |
| 1478 | MImpgal/d | Mega Imperial gallons per day |
| 1479 | µbbl/s | Microbarrels per second |
| 1480 | mbbl/s | Millibarrels per second |
| 1481 | kbbl/s | Kilobarrels per second |
| 1482 | Mbbl/s | Megabarrels per second |
| 1483 | µbbl/min | Microbarrels per minute |
| 1484 | mbbl/min | Millibarrels per minute |
| 1485 | kbbl/min | Kilobarrels per minute |
| 1486 | Mbbl/min | Megabarrels per minute |
| 1487 | µbbl/h | Microbarrels per hour |
| 1488 | mbbl/h | Millibarrels per hour |
| 1489 | kbbl/h | Kilobarrels per hour |
| 1490 | Mbbl/h | Megabarrels per hour |
| 1491 | µbbl/d | Microbarrels per day |
| 1492 | mbbl/d | Millibarrels per day |
| 1493 | kbbl/d | Kilobarrels per day |
| 1494 | Mbbl/d | Megabarrels per day |
| 1495 | $\mu m^3/s$ | Cubic micrometers per second |
| 1496 | $mm^3/s$ | Cubic millimeters per second |
| 1497 | $km^3/s$ | Cubic kilometers per second |
| 1498 | $Mm^3/s$ | Cubic megameters per second |
| 1499 | $\mu m^3/min$ | Cubic micrometers per minute |
| 1500 | $mm^3/min$ | Cubic millimeters per minute |
| 1501 | $km^3/min$ | Cubic kilometers per minute |
| 1502 | $mm^3/min$ | Cubic megameters per minute |
| 1503 | $\mu m^3/h$ | Cubic micrometers per minute |

| Value | Display | Description |
|---|---|---|
| 1504 | $mm^3/h$ | Cubic millimeters per minute |
| 1505 | $km^3/h$ | Cubic kilometers per minute |
| 1506 | $Mm^3/h$ | Cubic megameters per minute |
| 1507 | $\mu m^3/d$ | Cubic micrometers per day |
| 1508 | $mm^3/d$ | Cubic millimeters per day |
| 1509 | $km^3/d$ | Cubic kilometers per day |
| 1510 | $Mm^3/d$ | Cubic megameters per day |
| 1511 | $cm^3/s$ | Cubic centimeters per second |
| 1512 | $cm^3/min$ | Cubic centimeters per minute |
| 1513 | $cm^3/h$ | Cubic centimeters per hour |
| 1514 | $cm^3/d$ | Cubic centimeters per day |
| 1515 | $kcal_{th}/kg$ | Kilocalories per kilogram[1] |
| 1516 | $Btu_{th}/lb$ | British thermal units per pound[1] |
| 1517 | kL | Kiloliter |
| 1518 | kL/min | Kiloliters per minute |
| 1519 | kL/h | Kiloliters per hour |
| 1520 | kL/d | Kiloliters per day |
| 1551 | S/cm | Siemens per centimeter |
| 1552 | µS/cm | Microsiemens per centimeter |
| 1553 | mS/m | Millisiemens per meter |
| 1554 | µS/m | Microsiemens per meter |
| 1555 | $M\Omega \cdot cm$ | Megaohm centimeter[1] |
| 1556 | $k\Omega \cdot cm$ | Kiloohm centimeter[1] |
| 1557 | Weight% | Weight percent |
| 1558 | mg/L | Milligram per liter |
| 1559 | µg/L | Microgram per liter |
| 1560 | %Sat | - |
| 1561 | vpm | - |
| 1562 | %vol | Volume percent |
| 1563 | ml/min | Milliliters per minute |
| 1564 | $mg/dm^3$ | Milligrams per cubic centimeter |
| 1565 | mg/L | Milligram per liter |
| 1566 | $mg/m^3$ | Milligrams per cubic meter |
| 1567 | ct | Carat (jewels) = $200.0 \cdot 10^{-6}$ kg |
| 1568 | lb (tr) | Pound (troy or apothecary) = 0.3732417216 kg |
| 1569 | oz (tr) | Ounce (troy or apothecary) = 1/12 lb (tr) |
| 1570 | fl oz (U.S.) | Ounce (U.S. fluid) = (1/128) gal |
| 1571 | $cm^3$ | Cubic centimeter = $10^{-6}$ $m^3$ |
| 1572 | af | acre foot = 43560 $ft^3$ |
| 1573 | $m^3$ normal | Cubic meter |
| 1574 | L normal | Liter |

| Value | Display | Description |
|---|---|---|
| 1575 | $m^3$ std. | Standard cubic meter |
| 1576 | L std. | Standard liter |
| 1577 | ml/s | Milliliters per second |
| 1578 | ml/h | Milliliters per hour |
| 1579 | ml/d | Milliliters per day |
| 1580 | af/s | Acre foot per second |
| 1581 | af/min | Acre foot per minute |
| 1582 | af/h | Acre foot per hour |
| 1583 | af/d | Acre foot per day |
| 1584 | fl oz (U.S.)/s | Ounces per second |
| 1585 | fl oz (U.S.) /min | Ounces per minute |
| 1586 | fl oz (U.S.)/h | Ounces per hour |
| 1587 | fl oz (U.S.)/d | Ounces per day |
| 1588 | $m^3$/s normal | Standard cubic meters per second |
| 1589 | $m^3$/min normal | Standard cubic meters per minute |
| 1590 | $m^3$/h normal | Standard cubic meters per hour |
| 1591 | $m^3$/d normal | Standard cubic meters per day |
| 1592 | L/s normal | Standard liters per second |
| 1593 | L/min normal | Standard liters per minute |
| 1594 | L/h normal | Standard liters per hour |
| 1595 | L/d normal | Standard liters per second |
| 1596 | $m^3$/s std. | Standard cubic meters per second |
| 1597 | $m^3$/min std. | Standard cubic meters per minute |
| 1598 | $m^3$/h std. | Standard cubic meters per hour |
| 1599 | $m^3$/d std. | Standard cubic meters per day |
| 1600 | L/s std. | Standard liters per second |
| 1601 | L/min std. | Standard liters per minute |
| 1602 | L/h std. | Standard liters per hour |
| 1603 | L/d std. | Standard liters per day |
| 1604 | $ft^3$/s std. | Standard cubic feet per second |
| 1605 | $ft^3$/d std. | Standard cubic feet per day |
| 1606 | oz/s | Ounces per second |
| 1607 | oz/min | Ounces per minute |
| 1608 | oz/h | Ounces per hour |
| 1609 | oz/d | Ounces per day |
| 1610 | Paa | Pascal (absolute) |
| 1611 | Pag | Pascal (gauge) |
| 1612 | GPaa | Gigapasacal (absolute) |
| 1613 | GPag | Gigapascal (gauge) |
| 1614 | MPaa | Megapascal (absolute) |
| 1615 | MPag | Megapascal (gauge) |
| 1616 | kPaa | Kilopascal (absolute) |
| 1617 | kPag | Kilopascal (gauge) |

| Value | Display | Description |
|---|---|---|
| 1618 | mPaa | Millipascal (absolute) |
| 1619 | mPag | Millipascal (gauge) |
| 1620 | µPaa | Micropascal (absolute) |
| 1621 | µPag | Micropascal (gauge) |
| 1622 | hPaa | Hectopascal (absolute) |
| 1623 | hPag | Hectopascal (gauge) |
| 1624 | gf/cm$^2$a | |
| 1625 | gf/cm$^2$g | |
| 1626 | kgf/cm$^2$a | |
| 1627 | kgf/cm$^2$g | |
| 1628 | SD4°C | Standard density at 4°C |
| 1629 | SD15°C | Standard density at 15°C |
| 1630 | SD20°C | Standard density at 20°C |
| 1631 | PS | Metric horsepower |
| 1632 | ppt | Parts per trillion = $10^{12}$ |
| 1633 | hl/s | Hectoliters per second |
| 1634 | hl/min | Hectoliters per minute |
| 1635 | hl/h | Hectoliters per hour |
| 1636 | hl/d | Hectoliters per day |
| 1637 | bbl (liq)/s | Barrels (US liquid) per second |
| 1638 | bbl (liq)/min | Barrels (US liquid) per minute |
| 1639 | bbl (liq)/h | Barrels (US liquid) per hour |
| 1640 | bbl (liq)/d | Barrels (US liquid) per day |
| 1641 | bbl (fed) | Barrel (U.S. federal) = 31 gallons |
| 1642 | bbl (fed)/s | Barrels (US federal) per second |
| 1643 | bbl (fed)/min | Barrels (US federal) per minute |
| 1644 | bbl (fed)/h | Barrels (US federal) per hour |
| 1645 | bbl (fed)/d | Barrels (US federal) per day |
| 1998 | Unknown unit | To be used when the unit of measure is not known during configuration |
| 1999 | Special | Special units |
| | | |

[1] A notation different to the PA profile must be used in order to represent this unit in conformity with the system.

[2] "m2" is used instead of "m3" in the Chinese and Japanese documentation.

## 2.1.13 Consumption of Process Objects

### Process Objects (PO) Count

All function blocks of the APL uses in general one PO. Exceptions are listed in the table:

| Block type | PO count |
|---|---|
| AV | 0 |
| Evt16Ts | 0 |
| EventTs | 0 |
| KalFilt | 50 |
| ModPrCon | 50 |
| MPC10x10 | 100 |
| PrDrpMon | 5 |
| PST | 15 |
| PumpMon | 10 |
| SteadySt | 1 |
| VlvMon | 5 |
| VlvPosL | 2 |

## 2.1.14 Resynchronization

With a resynchronization the control outputs and the state of a block are adapted to a running process. This is acheived by tracking the control outputs for one cycle through the actual feedback signals (FbkXxx) or readback values (Rbk, PV_In).

The resynchronization is enabled over feature bits.

**Resynchronization after CPU Startup:**

| | |
|---|---|
| Feature Bit17 =1 | OpDi01, OpDi03, OpAnL |
| Feature2 Bit17 =1 | MotL, MotRevL, MotSpdL, MotSpdCL, VlvL, VlvMotL, VlvAnL, VlvPosL |

In the first cycle after OB100 commands, block state and control outputs are tracked through the feedback signals and readback values from the process. If they are not available or plausible, the controls are put on hold.

**Resynchronization after Out of service:**

| | |
|---|---|
| Feature Bit18 =1 | OpDi01, OpDi03, OpAnL |
| Feature2 Bit18 =1 | MotL, MotRevL, MotSpdL, MotSpdCL, VlvL, VlvMotL, VlvAnL, VlvPosL |

In the first cycle after leaving "Out of service" commands, block state and control outputs are tracked through the feedback signals and readback values from the process. If they are not available or plausible, the actual position is not changed. In the standard view and preview the operator can see which state the block would assume after leaving "Out of service" mode.

**Resynchronization after Simulation:**

| | |
|---|---|
| Feature2 Bit18 =1 | MotL, MotRevL, MotSpdL, MotSpdCL, VlvL, VlvMotL, VlvAnL, VlvPosL |

In the first cycle after leaving simulation function commands, block state and control outputs are tracked through the feedback signals and readback values from the process. If they are not available or plausible, the actual position is not changed. In the standard view and preview the operator can see which state the block would assume after leaving simulation function.

**Resynchronization after CSF:**

| | |
|---|---|
| Feature2 Bit19 =1 | MotL, MotRevL, MotSpdL, MotSpdCL, VlvL, VlvMotL, VlvAnL, VlvPosL |

In the first cycle after leaving control system fault (CSF 1->0) commands, block state and control outputs are tracked through the feedback signals and readback values from the process. If they are not available or plausible, the actual position is not changed. In the standard view and preview the operator can see which state the block would assume after leaving control system fault.

**Resynchronization with input StateTrkOn:**

| | |
|---|---|
| Feature Bit20 | OpDi01, OpDi03, OpAnL |
| Feature2 Bit20 =1 | MotL, MotRevL, MotSpdL, MotSpdCL, VlvL, VlvMotL, VlvAnL, VlvPosL |

After a trigger (0->1) at StateTrkOn input commands, block state and control outputs are tracked through feedback signals and readback value. If they are not available or plausible, the actual position is not changed. In the standard view and preview the operator can see which state the block would assume after a trigger of StateTrkOn input.

**General restrictions of resynchronization:**

In many cases the control outputs are not tracked from the feedback signals or readback values:

- **Active commands** in automatic, local mode or forcing:
  - XxxAut =1 in button mode (Feature Bit4 =0)
  - XxxLocal =1 in local button mode (Feature2 Bit4 =1)
  - In Switch mode (Feature Bit4 =1, Feature2 Bit4 =1) the commands are always active and cannot be tracked.
- **Interlocks:** Tracking of an interlocked control is not possible. On the other hand the resynchronization can set the block to a position where an interlock permission is active. Or in case of an interlock (not protected), regarding the direction of the resynchronization can be set to a none interlocked position.
- **Active errors** like monitoring errors with MonSafePos =1, FaultExt =1, CSF =1 with error behavior Feature Bit18 =1, Trip =1: Tracking of a control is not possible. But the state/position can be adapted by resynchronization.
- **Rapid Stop**: Tracking of a control is not possible. But the state/position can be adapted by resynchronization.

Tracking of state is independent from the interlock permission

# 2.2 Functions of the block icons

## 2.2.1 Block icon structure

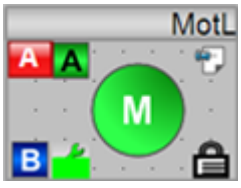
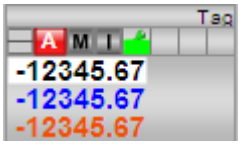
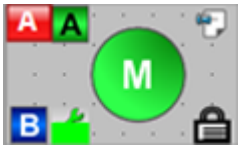
### Block icon structure

The new block icons are located in the template "@PCS7TypicalsAPLV9.PDL" and "@TemplateAPLV9.PDL".

**Note**

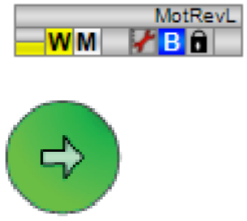
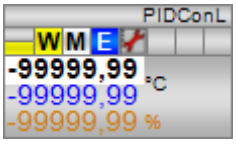
@PCS7TypicalsAPLV7.PDL and @TemplateAPLV7.PDL is no longer supplied with the Setup

There are two types of block icons (V9.0), those with a display of the instance-specific name and those without:

| | |
|---|---|
|  | Block icon of MotL with instance-specific name |
|  | Block icon of PIDConL with instance-specific name |
|  | Block icon of MotL without instance-specific name The toolbar only shows the information that is actually available. |
|  | Block icon of PIDConL without instance-specific name The toolbar only shows the information that is actually available. |

The old block icons are located in the template "@PCS7TypicalsAPLV7.pdl" and "@TemplateAPLV7.PDL".

There are two types of block icons (V7.1), those with a display of the instance-specific name and those without:

| | |
|---|---|
|  | Block icon of MotRevL with instance-specific name |
|  | Block icon of PIDConL with instance-specific name |
|  | Block icon of MotRevL without instance-specific name The toolbar only shows the information that is actually available. |
|  | Block icon of PIDConL without instance-specific name The toolbar only shows the information that is actually available. |

You can select one of these block icons. See section Configuring the block icons (Page 267) for more information.

A block icon has several display areas:

- CPU stop
- Instance-specific name
- Icon for the block
- Analog value display
- Status bar for the block status

**NoteView documents**

You can configure a maximum of three files per block symbol instance, which can be selected in the memo view during the process control and are displayed in a separate window.

These files must be located in the GraCS folder in the NoteView subdirectory.

The file names are configured at the block icon at the attributes NoteView1, NoteView2 and NoteView3. Only .mht file formats are supported and the file extension ".mht" must be given (for exmaple, " File1.mht") at the configuration.

**Note**

1. A maximum of three files are configured per block instance.
2. Only .mht file formats are permitted.
3. The file extension *.mht must be given at the configuration.
4. In multi-projects, all relevant files are stored in the path of the server project (OS project path >> GraCS >> NoteView >> (files in MHT format)).
5. If the files are not configured at the block icon, the file selection and the "Open" button are deactivated in the memo view (that is, grayed out).
6. If the file formats other than .mht files are configured, the error message "Invalid file format" is displayed when the file is opened.
7. The maximum allowed file size is 25 MB. If a file exceeds this size, the error message "File size exceeded" is displayed when the file is opened.

## Displaying CPU stop

With a CPU stop, boxes are unavailable and a yellow warning triangle is displayed in the group display for blocks with messaging.

## Instance-specific name

The name of the associated block is shown in the instance-specific name, for example for the PIDConL block:




You can change this name in the object properties of the instance block.

There are block icons with or without display of the instance-specific name. Refer to the individual block descriptions to learn about them.

You can reach the visible display for blocks without display of the instance-specific name in two different ways:

- **Displaying individual instance-specific names**: Click on the block icon while holding down the Shift key: The name remains visible as long as the process picture is displayed.
- **Displaying all instance-specific names at once**: All instance-specific names can be made visible in a process picture at once by clicking a button. To do this, copy this button into the process picture of the chart from the @TemplateAPLV9.PDL or insert this button in the "button area" of WinCC. If you insert it into the button area, read the manual section "PCS 7 OS Process Control " > "Layout of the User Interface".
  The instance-specific names are hidden once more by clicking the button again.

## Icon for the block

Block icons for technologic blocks have their own operable symbols (for example, for the MotRevL block) and represent a status display of the block.



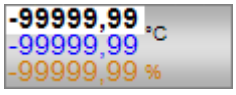
This symbol can be positioned at various locations, 0°, 90°, 180° and 270°. Refer to the Configuring the block icons (Page 267) section for more on this. You can change the operating mode of the block by right-clicking on the status display. Refer to the Operation via the block icon (Page 269) section for more on this.

**Note**

Block icons for motors and valves are available in various mounting positions and forms of representation. However, a general state icon is always displayed in the faceplates that does not consider the mounting position and form of representation in the block icons.

## Analog value display

For block icons with analog value displays (for example, for the PIDConL block), there are basic settings that differ depending of the associated unit of measure:




Refer to the Configuring the block icons (Page 267) section for more on this.

These analog values can be controlled according to the Operator control permissions (Page 285). See also the section Operation via the block icon (Page 269) for more on this.

As of V9.0 block icons, when the actual values are displayed with limit monitoring in the analog value displays, the background and font colors for an alarm, warning or tolerance correspond to the message colors.

**Note**

As of V9.0 block icons, when the actual values are displayed with limit monitoring in the analog value displays, the background and font colors for an alarm, warning or tolerance, control system fault or control system fault correspond to the message colors.

If the analoge value display shows *****, the display field is too small for the parameterized format specification. In this case, adapt the format specification AnalogValueFormatx in the block icon APL_BLOCK_ICON.

For specification of the format, refer to Decimal points for the analog value display (Page 277).

### Status bar for the block status

The status bar of the block icon provides an overview of the overall status of the block (see figure below).




The arrangement of icons in the following tables is prioritized from high to low.

The following elements can be displayed:

### Alarms, warnings, tolerances, and messages

Refer to the Monitoring functions in the Advanced Process Library (Page 92) section for more on this.

| Icon | Meaning |
|---|---|
|  | No messages are output. |
|  | A fault has occurred. |
|  | An alarm is triggered. |
|  | An alarm for the high alarm limit is triggered. |
|  | An alarm for the low alarm limit is triggered. |
|  | An warning is triggered. |
|  | A warning for the high warning limit is triggered. |
|  | A warning for the low warning limit is triggered. |
|  | A tolerance violation has occurred. |
|  | A tolerance violation for the high tolerance limit has been triggered. |
|  | A tolerance violation for the low tolerance limit has been triggered. |
|  | The block has an operator prompt. |
|  | There is a process status determination. |

**Note**

The message classes Alarm, Warning and Tolerance are not valid for user-configured message classes. Please take into consideration the validity of terms for user-configured message classes (Page 45).

**Note**

For user-configured message classes, the symbols for high/low alarm, warning and tolerance limits are displayed only for limit violations which are triggered within the APL blocks with the colors of the user-configured message classes.

## Operating modes

Refer to the Overview of the modes (Page 76) section for more on this.

| Icon | Meaning |
|---|---|
|  | The block is in automatic mode. |
|  | The block is in the "On" operating mode. |
|  | The block is in manual mode. |
|  | The block is in program mode (only for controllers). |
|  | The block is in local mode. |
|  | The block is in the "Out of service" operating mode.<br>In this operating mode, no other symbols are displayed and no values are shown in the analog value display except for:<br>- Display for an active message in the memo view<br>- Display for the maintenance enable |

**Note**

No symbol is displayed for the "On" operating mode (no green "O" displayed) if the block only has the operating modes "On" and "Out of service".

## Internal or external setpoint

Refer to the section Setpoint specification - internal/external (Page 136).

| Icon | Meaning |
|---|---|
|  | Setting the setpoint internally |
|  | External setpoint specification |
|  | SP/MV ramp active |

## Signal status

Refer to the Forming and outputting signal status for blocks (Page 116) section for more on this.

| Icon | Meaning |
|---|---|
|  | The block is in simulation. |
|  | Signal status is "Bad, device related". |
|  | Signal status is "Bad, process related". |
|  | Signal status is "Uncertain, device related". |
|  | Signal status is "Uncertain, process related". |
|  | A maintenance request is pending. |
|  | Block is released for maintenance |

## Tracking and forcing of values and bypasses

See sections Forcing operating modes (Page 45), Interlocks (Page 107) and Manual and automatic mode for control blocks (Page 79).

| Icon | Meaning |
|---|---|
|  | At least one value has been forced |
|  | Value is tracked |
|  | There is a bypass of an interlock or a bypass condition of an upstream Intlock FB or tag bypass is active. |

## Interlocks

Refer to the Interlocks (Page 107) section for more on this.

| Icon | Meaning |
|---|---|
|  | Block is not interlocked. |
|  | Block is interlocked. |
|  | Bypass protection. |
|  | Rapid stop is commanded. |

### Memo display

| Icon | Meaning |
|---|---|
|  | A message is available in the memo view. |

### Customer-specific icons

In the block icons, the icons can be replaced with customer-specific icons. A subdirectory must therefore be created in the OS project directory "GraCS" and the customer-specific icons must be inserted there with the same name as before. The name of the subdirectory must be adapted in the block icon in the "**PictureDirectory**" property ("Configurations" group). The icons in the subdirectory are not changed by the OS project editor or a new version of APL faceplates.

**Note**

- The "PictureDirectory" property is an instance-specific property. This means that the adaptations of block icons already configured in the process pictures are not changed by an update of the block icons in the process picture.
- We recommend that you adapt this property in the process picture and not in the "@PCS7TypicalsAPLVx.pdl" or "@TemplateAPLVx.PDL" pictures. The "@" pictures are system pictures and are replaced by the new version of APL faceplates. Customer changes will be lost.
- In the WinCC Graphics Designer, the PCS 7 "Export Objects" wizard can be used to generate a list of adaptations of instance-specific properties in the block icons. The "PictureDirectory" property can be changed in the exported list and collectively transferred back into the block icons with the PCS 7 "Import Objects" wizard.
- It is possible to use several subdirectories. This can be useful to create different variants of block icons in the process pictures.

### See also

Motor protection function (Page 106)

Forming the group status for interlock information (Page 113)

User-configured message classes (Page 45)

## 2.2.2 Configuring the block icons

### Configuring the block icons

There are two ways to configure your block icons:

- Automatically
- Manually

The new block icons are in the template picture "@PCS7TypicalsAPLV9.PDL" and "@TemplateAPLV9.PDL".

**Note**

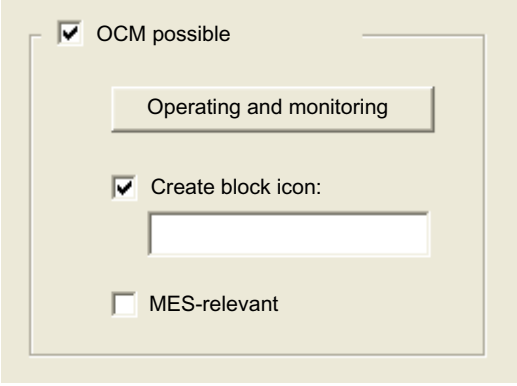
@PCS7TypicalsAPLV7.PDL and @TemplateAPLV7.PDL is no longer supplied with the Setup.

If you still use the V7 block icons in your project and they should continue to be used, the V7 template "@PCS7TypicalsAPLV7.PDL" must be present in the OS project directory (GraCS). In addition, you must delete the newer template images "@PCS7TypicalsAPLV9.PDL" or "@PCS7TypicalsAPLV8.PDL" from the OS project directory.

Deselect also the update for the V9 template in the OS project editor on the basic data tab.

### Automatic configuration of block icons

There is a variety of block icons, which you can select for a block. You select a block icon by entering the number of the block icon in the field "OCM possible" > "Create block icon" in the object properties of the block instance:




You can find the names (numerical entry) of the respective block icons in the description for the blocks (Operator Control and Monitoring section).

If you do not enter a number for a block icon, the number 1 is always used for the block icon. The template for automatically generated block icons is @PCS7TypicalsAPLV9.PDL.

### Additional information

- Manual *Process Control System PCS 7; Engineering System*

### Manual configuration of block icons

You can configure block icons manually by copying them from the @TemplateAPLV7.PDL/ @TemplateAPLV9.PDL template and inserting them into plant pictures.

The connection to the process tag is established with the "Connect faceplate with process tag" Wizard, see WinCC Information System, section "Making Process Pictures Dynamic" and "Standard Dynamics".

You can find information on exporting/importing and updating these objects in the WinCC Information System, section "Graphic Object Update Wizard". Use the "TemplateControlAPL.cfg" configuration file for these wizards.

**Note**

- The procedure for changing the tooltip text of the block icons is described in the *APL Style Guide*. Otherwise, this property may not be changed.

## 2.2.3 Operation via the block icon

### Operation via the block icon

The block icon can be used to operate elements directly if a relevant operator control permission (`OS_Perm`) is available. This operator control permission can be configured in the engineering system (ES).

The operation is performed by right-clicking on the element involved. The operable elements in the block icon include:

- Switching the operating mode
- Internal and external setpoint specification
- Changing the process value, setpoint and manipulated variable
- Changing the operating state

The operation is then performed in the same way as in the faceplate. Refer to the section Switching operating states and operating modes (Page 289) as well as Changing values (Page 291).

## 2.2.4 Block icons for PID and FM controller

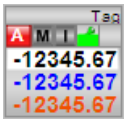
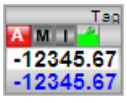
### Block icons for PID and FM controller

A variety of block icons are available with the following functions:

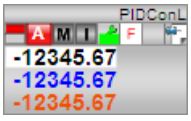
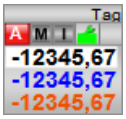
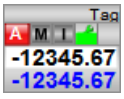
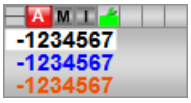
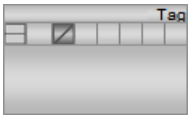
- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits (tolerance limits not with PIDConS) as well as control system faults
- Operating modes
- Internal and external setpoint specification
- Signal status, release for maintenance
- Memo display

- Process value (black, with and without decimal places, with and without color change at limit violations)
- Setpoint (blue, with decimal places)
- Feedback value (red, with decimal points), not available for types 3 and 4
  With FmCont and FmTemp as a step controller without feedback, there is no feedback value
- "Ramp active" indication is not available in FMCont and FMTemp blocks.
- only `PIDStepL`: Position feedback value (green, with decimal points), not available for types 3 and 4
- Interlock without reset (only for PIDConR and PIDConL)

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in the CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | Block icon in the full display |
|  | 6 | Block icon in the full display |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | |
|  | 6 | |
|  | 7 | |
|  | 8 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |
| Special case for `PIDStepL`: | | |
|  | - | In the case of the `PIDStepL` block only, the green value shows the position feedback (only visible if `WithRbk = 1` has been assigned parameters). With operation by means of the block icon, however, the manipulated variable is operated. |

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

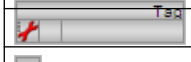
## 2.2.5 Block icon for interlock blocks

### Properties of the block icon for interlock blocks Intlk02, Intlk04, Intlk08, Intlk16

A variety of block icons are available with the following functions:

- Signal status
- Memo display
- Output signal

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Block icon for |
|---|---|---|
|  | 1 | Intlk02 |
|  | 1 | Intlk04 |
|  | 1 | Intlk08 |
|  | 1 | Intlk16 |
|  | 2 | Intlk02 |
|  | 2 | Intlk04 |
|  | 2 | Intlk08 |
|  | 2 | Intlk16 |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Block icon for |
|---|---|---|
|  | 1 | Intlk02 |
|  | 1 | Intlk04 |
|  | 1 | Intlk08 |

| Icons | Selection of the block icon in CFC | Block icon for |
|---|---|---|
|  | 1 | Intlk16 |
|  | 2 | Intlk02 |
|  | 2 | Intlk04 |
|  | 2 | Intlk08 |
|  | 2 | Intlk16 |
|  | 3 | Intlk02 |
|  | 3 | Intlk04 |
|  | 3 | Intlk08 |
|  | 3 | Intlk16 |

## Block icons (mini) without display of instance-specific names

These icons only show the output signal. They take the form of a small rectangle.

## Display of the output signal

The display can show the following states for the output signal (priority from high to low):

- Gray: No inputs interconnected at the interlock block, the block is not used
- Blue: The output signal is 1, at least one input signal is bypassed. There is no gray state.
- Yellow: The signal status is 16#60; the output signal is simulated. There are no gray and blue states.
- Red: The output signal is 0; there is an interlock. There are no gray, blue, and yellow states.
- Green: The output signal is 1; the block is in the good state. There are no gray, blue, yellow, and red states.

Additional information on the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

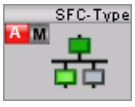
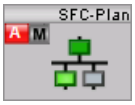
## 2.2.6 Block icons for SFC

### Block icons for SFC

A variety of SFC block icons are available with the following functions:

- Alarm indicator
- Current active mode indicator ("Manual" or "Automatic" mode)
- Operator request icon
- Indicator for SFC operating status

The block icons from template @PCS7TypicalsAPLV9.PDL:

| Block icons | Type 2 | Type 3 |
|---|---|---|
| SFC-Type |  |  |
| SFC-Plan |  |  |

The following icons are dispayed in the SFC block icon:

1. Mode indicators:

| Icon | Bit number | Tag name | Description |
|---|---|---|---|
|  | 7 | .SFC_STATE | The block is in "Automatic" mode. |
|  | 7 | .SFC_STATE | The block is in "Manual" mode. |

2. Status indicators:

| Icon | Bit number | | | | Tag name | Description |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | | |
|  | 0 | 0 | 0 | 0 | .SFC_STATE | IDLE |
|  | 0 | 0 | 0 | 1 | .SFC_STATE | RUN |
|  | 0 | 0 | 1 | 0 | .SFC_STATE | COMPLETED |
|  | 0 | 0 | 1 | 1 | .SFC_STATE | HELD |
|  | 0 | 1 | 0 | 0 | .SFC_STATE | ABORTED |
|  | 0 | 1 | 0 | 1 | .SFC_STATE | STARTING |
|  | 0 | 1 | 1 | 0 | .SFC_STATE | COMPLETING |
|  | 0 | 1 | 1 | 1 | .SFC_STATE | ERROR_COMPLETING |
|  | 1 | 0 | 0 | 0 | .SFC_STATE | HOLDING |
|  | 1 | 0 | 0 | 1 | .SFC_STATE | RESUMING |
|  | 1 | 0 | 1 | 0 | .SFC_STATE | ERROR |
|  | 1 | 0 | 1 | 1 | .SFC_STATE | HELD_ERROR |
|  | 1 | 1 | 0 | 0 | .SFC_STATE | RESU_ERROR |
|  | 1 | 1 | 0 | 1 | .SFC_STATE | ABORTING |
|  | 1 | 1 | 1 | 0 | .SFC_STATE | STOPPING |
|  | 1 | 1 | 1 | 1 | .SFC_STATE | STOPPED |

3. Acquired status indicators (for SFC type only):

| Icon | Bit number | Tag name | Description |
|---|---|---|---|
|  | 14 | .VSTATUS | EM-SFC is acquired by a superordinated equipment phase (EPH) |
|  | 15 | .VSTATUS | Acquiring of at least one subordinated equipment module (EM) failed |

**See also**

Block icon structure (Page 260)
Configuring the block icons (Page 267)
Operation via the block icon (Page 269)

## 2.2.7 Adding block icons to static picture components

### Static picture component for the block icons

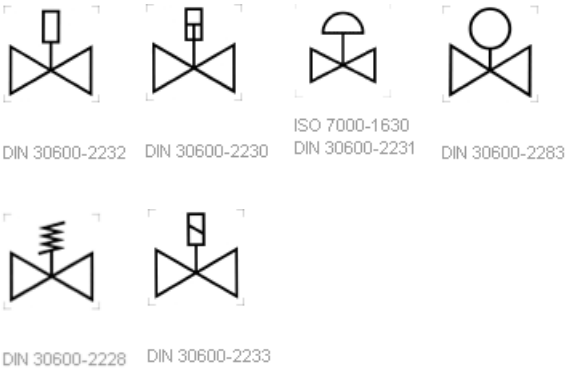
The block icons for drives and valves contain no static picture components. However, you can add static picture components to the block icons by copying them from the @TemplateAPLV9.PDL/@TemplateAPLV7.PDL template and placing them over the block icons.

The following static picture components are available:

- Static picture components for motor blocks:




- Static picture components for valve blocks:




**Note**

The static picture components for the valves are visible here above the valve icon.

Note that all the block icons of the APL are located on layer 0 of the process picture. Layer 1 is intended for the static picture component. This ensures that the static picture component is always over the block icon. If you have changed this layer of the block icon, the static picture component should always be placed in a higher layer.

## 2.2.8 Decimal points for the analog value display

The decimal points for the analog value display can parameterize centrally in the block icon at the attribute Configurations > AnalogValueFormat1...AnalogValueFormat5. If the attribute is empty, the format specification will be taken directly from the faceplate PDL file.

The parameterization in the block icon takes effect in the standard view, limit view, and the block icon. The following table shows the analog values for which format specification can be parametrized in the block icon:

| Blocks | AnalogValue-Format1<br><br>Standard view, Limit view | AnalogValue-Format2<br><br>Standard view, Limit view | AnalogValue-Format3<br><br>Standard view, Limit view | AnalogValue-Format4<br><br>Standard view | AnalogValue-Format5<br><br>Limit view for motors |
|---|---|---|---|---|---|
| **MotL** | | | UserAna1 | UserAna2 | AV values |
| **MotRevL** | | | UserAna1 | UserAna2 | AV values |
| **MotSpdL** | | | UserAna1 | UserAna2 | AV values |
| **MotSpdCL** | Rbk values<br>ER values | SP values | UserAna1 | UserAna2 | AV values |
| **VlvL** | | | UserAna1 | UserAna2 | - |
| **VlvMotL** | | | UserAna1 | UserAna2 | AV values |
| **VlvPosL** | Rbk values<br>MV value | | UserAna1 | UserAna2 | AV values |
| **VlvAnL** | Rbk values<br>MV value | | UserAna1 | UserAna2 | - |
| **Vlv2WayL** | | | UserAna1 | UserAna2 | - |
| **DoseL** | PV values | SP values | UserAna1 | UserAna2 | - |
| **MonAnL** | PV values | | UserAna1 | UserAna2 | - |
| **MonAnS** | PV values | | | | - |
| **OpAnL** | PV values | SP values | | | - |
| **OpAnS** | PV values | SP values | | | - |
| **PIDConL** | PV values | SP values | UserAna1<br>MV values<br>Rbk values | UserAna2 | - |
| **PIDConS** | PV values | SP values | MV values<br>Rbk values | | - |
| **PIDConR** | PV values | SP values | UserAna1<br>MV values<br>Rbk values | UserAna2 | - |
| **PIDStepL** | PV values | SP values | UserAna1<br>MV values<br>Rbk values | UserAna2 | - |

| Blocks | AnalogValueFormat1...AnalogValueFormat4 | AnalogValueFormat5...AnalogValueFormat8 | AnalogValueFormat9 |
|---|---|---|---|
| **ModPreCon** | CV1...CV4, SP1...SP4 values | MV5...MV8 values | DV values |

| Blocks | AnalogValueFormat1...AnalogValueFormat10 | AnalogValueFormat11...AnalogValueFormat20 | AnalogValueFormat21...AnalogValueFormat24 |
|---|---|---|---|
| **MPC10x10** | CV1...CV10, SP1...SP10 values | MV1...MV10 values | DV1...DV5 values |

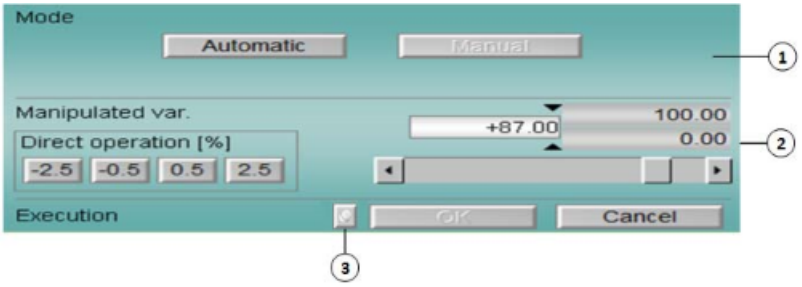
**Note**

If the analoge value display shows *****, the display field is too small for the parameterized format specification. In this case, adapt the format specification AnalogValueFormatx in the block icon APL_BLOCK_ICON.

# 2.3 Functions of the faceplates

## 2.3.1 Fast Operation for PID Controllers

### Fast Operation for PID Controllers




The concerned blocks are:

- PIDConL
- PIDStepL
- PIDConR

### Enabling Fast Operation

When internal tag "@APLFastOperationEnable" is set to 1, the above mini faceplate can be opened from the block icon. You can right-click on the mode and the manipulated variable fields on block icon will open this faceplate.

1. **Changing modes**
   Only two modes - automatic and manual are available.You can click on the respective buttons to change the mode.
2. **Changing manipulated variable**
   Manipulated variable can be changed by either entering the value directly through slider or by direct operation buttons. Both 2 and 3 step operation are available when changing manipulated variable.
3. **Pin**
   Faceplate is always pinned by default. Pin button is not operable and therefore faceplate can not be unpinned.

## 2.3.2 Structure of the faceplate

### General functions of the faceplates

This section provides general information that applies to all faceplates.

### Recommended screen resolution

The faceplates are shown in full on the screen with a resolution of 1280 x 1024. The full-screen mode (press the F11 key) must also be activated on the Web client.

### Scaling the faceplates

You can scale the faceplates from 50% to 200%. The value to scale is given in the internal variable "@APLFaceplateScaleFactor" which is of type unsigned 32-bit value. The default value of the scale is 100. You can scale the faceplates from 101% to 200% by providing a value in the range of 101 to 200 in the internal variable. Similarly, you can scale down the faceplates from 50% to 99% by providing a value in the range of 50 to 99. The reference for calculation is default size of the faceplates which is 100%. This means if the value given is 150, the faceplates will be scaled to 150% of the default size. Similarly, if the value given is 90, the faceplate will be scaled down to 90% of the default size.

The faceplates can also be scaled up or down using the following buttons in a process picture. These buttons are contained in @TemplateAPLV9.PDL and should be configured for both OS-server and OS-client.

| Button | Name | Description |
|---|---|---|
|  | BtnIncreaseZoom | Increase the zoom factor by 5% |
|  | BtnDecreaseZoom | Decrease the zoom factor by 5% |
|  | BtnResetZoom | Reset the zoom factor to 100% |
|   | ZoomValue | Displays the current Zoom value |

**Note**

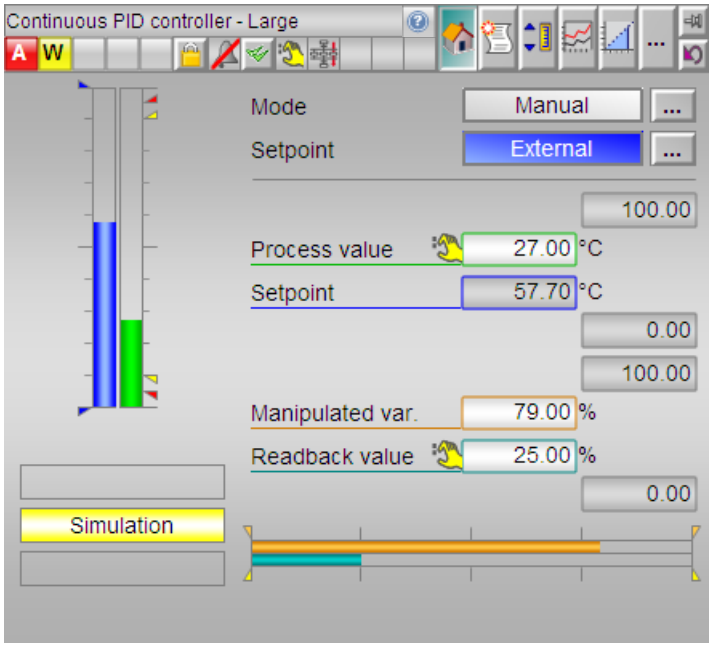
You must reopen the faceplate after changing the scale value. Already opened faceplate will not be scaled if the scale value is changed.

### Displaying CPU stop

With a CPU stop, boxes are unavailable or hidden and a yellow warning triangle is displayed in the group display for blocks with messaging. No operations are possible.

## Opening the faceplate

Click on the block icon in WinCC to open the faceplate with the standard view; in this example this is the standard view of PIDConL:




**Note**

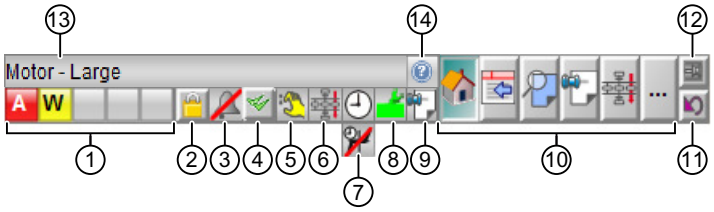
Some of the illustrations of the faceplate views and block icons in the help for the PCS 7 Advanced Process Library are examples or offline representations. The representations on runtime may vary.

The views differ depending on the block functions. All blocks that have faceplates provide a status bar where you can see the most important information relating to the block status. There are additional functions available that are described in the next sections.

**Note**

Display elements of block inputs with `16#FF` as their status default, are only shown when their block inputs are interconnected (status ≠ FF). Exception: The values for `Gain`, `TI` and `TD` are always displayed for controller blocks.

## Displays and operator controls




The faceplate provides the following display and operator controls:

(1) Group display

(2) Lock/unlock messages

(3) Suppress messages / alarm delays with the icon 

(4) Acknowledge alarms

(5) Worst signal status

(6) Batch display

(7) At least one delay time effective at the block

(8) Maintenance request and release

(9) Memo display

(10) Open views of the block

(11) Back to block icon

(12) Pin faceplate

(13) Instance name of the block

(14) Help button

## (1) Group display

The group display shows the information that is transferred from ALARM_8P of the block instance to WinCC.

- Alarms
- Warnings
- Tolerances
- Faults
- Operator prompts
- Process messages

**Note**

The message classes Alarm, Warning and Tolerance are not valid for user-configured message classes. For user-configured message classes, message types depend on the project-specific setting. Please take into consideration the validity of terms for user-configured message classes (Page 45).

### (2) Lock/unlock messages

The Lock/unlock messages button is used to lock/unlock the triggering of messages for all single control units in the process picture affected by group displays directly in the automation system.

### (3) Suppress messages

Message suppression indicates whether or not the "Suppress process messages" function in the AS block is activated with the `MsgLock` parameter. If message suppression is activated, all messages in this block instance – except for process control messages – are suppressed.

### (4) Acknowledge alarms

You can acknowledge all alarms from the block instance using this button.

You can hide this button from specific users / user groups using the permissions in PCS 7-OS. Refer to the *Process Control System PCS 7; OS Process Control* documentation for more on this.

### (5) Display for worst signal status

This display shows the worst signal status currently present. You will find more detailed information in section Forming and outputting signal status for blocks (Page 116).

### (6) Batch display

The batch display shows whether or not the block instance is in use by SIMATIC BATCH.

You will find more detailed information in section SIMATIC BATCH functionality (Page 73).

### (7) At least one delay time effective at the block

| | |
|---|---|
|  | This display shows you the active delay time. You will find more detailed information in section Display of delay times (Page 288). |
|  | The "dead band" function is temporarily deactivated. |

### (8) Maintenance request and release

This display shows you if a maintenance request or release has been made for this block.

You will find more detailed information in section Release for maintenance (Page 70).

### (9) Memo display

This display shows you if a message has been left in the memo view for you. You will find more detailed information in section Memo view (Page 338).

### (10) Open views of a block

You can use this field to open the various views of a block. Refer to the block description to learn of the available views. Left clicking shows the view in the same window. Right clicking opens a new window.

You can select from the following typical views here:

| Icon | Identifier |
|---|---|
|  | Standard view |
|  | Message view |
|  | Limit view (several limit views within a block are possible) |
|  | Trend view |
|  | Ramp view |
|  | Parameter view (several parameter views within a block are possible) |
|  | Preview |
|  | Memo view |
|  | Batch view |
|  | Setpoint view |
|  | Parameter view 1 (continues to parameter view 4, see line below) |
|  | Parameter view 4 |
|  | Quantity view |
|  | Flow view |
|  | Views (additional views within a block are possible) |

**Note**

The buttons are unavailable when views cannot be selected.

### (11) Back to block icon

Use this button to return to the block icon in the process image of the corresponding faceplate. You can use this function, for example, when you have pinned a block **(12)** and the process picture has changed in the meantime.

### (12) Pin faceplate

You can pin the faceplate on top of the user interface using this button. This allows you to change to another picture or area without closing the faceplate.

**Note**

You can learn about additional operator controls in the descriptions of the individual blocks.

### (13) Instance name of the block

This area displays the instance specific name of the block.

### (14) Help button

Click this button to open the *APL Operator Guide* online help of the corresponding view of the faceplate.

## 2.3.3 Operator control permissions

### Operator control permissions for blocks

The following conditions have an effect on operator control permissions:

- User management in the PCS 7 OS
- Local operating permission using the OpStations block
- Dependencies on operating modes in blocks
- Permissions via parameter assignment / interconnection of blocks
- Permissions via the `OS_Perm` input parameter at the block itself

### User management in the PCS 7 OS

The following operator authorization levels from user management are used in the APL:

- Process control (for example, manual/automatic mode switchover, changing setpoints and manipulated variables).
  With this operator control permission, operations can be performed in the standard view of all blocks and input can be made in the ramp and memo views. The "Out of service" operating mode cannot be used with process controlling.
- Higher process controls (for example, changing limits, controller parameters and monitoring times).
  With this operator control permission, all operations in all views of all blocks are possible, including the operations for "Out of service" operating mode. Exception: The operations listed under "Highest process controlling".
- Highest process controls (simulate process values and release process tag for maintenance).
  With this operator control permission, simulation can be switched on and off in the parameter view and the process tag for maintenance work can be released.
- Extended operation 1
  Free project-specific operator authorization
- Extended operation 2
  Free project-specific operator authorization

**Note**

Exceptions to the uses described above are listed in the descriptions of the individual views.

Each operation is assigned with an operator authorization level in the faceplates. This fixed assignment can be changed for each instance at the "operator authorization level" property of an I/O in the AS block (for example, SP_Int with PIDConL). The following assignment applies:

| Operator authorization level in the user management | Value "Operator authorization level" property |
|---|---|
| Process controls | 1 |
| Higher process controls | 2 |
| Highest process controls | 3 |
| Extended operation 1 | 4 |
| Extended operation 2 | 5 |

**Note**

- The free three assignments in the upper table can be changed in the block icon at the properties "OperationLevel1_backup", "OperationLevel2_backup" and "OperationLevel3_backup". Any operator authorization level from the user management can be assigned the values 1 to 3. This type of instance-specific configuration is still available only for reasons of downward compatibility and should no longer be used in new projects.
- The controls for the message system (e.g. acknowledge messages) and trend display (e.g. export) are fixed across the system and cannot be changed via the AS block.
  The "Lock messages" control can be changed per OS Server/ Station via the internal variable "`@LockMessageAuthLevel`" with the value of the operator authorization level from the user management (for example, value "6" for "Higher process controls").

### Local operating permission using the OpStations block

Local operating permission is an upstream operator control permission which is determined before the operator control permissions for user management and the release of the block, and is realized via the OpStations (Page 457) block.

If local operating permission is missing, operation of a block instance on an OS is usually blocked. Otherwise, when local operating permission is allowed, the operator control permission is normally determined through user management and the block.

Local operating permission can be set for each specific instance; in other words, block instances can be enabled or disabled for use on an operator station independently of one another.

You can find additional information for the use of local operating permission in the section Description of OpStations (Page 457).

### Dependencies on operating modes in blocks

You can execute various functions depending on the block mode. These permissions are stored in the block algorithm and are determined dynamically in online mode.

### Permissions via parameter assignment / interconnection of blocks

The block is either controlled by the operator or by the controller, depending on parameter settings or on the interconnection. An example for this is the switchover from manual to automatic mode by a higher-level controller or by the operator. These permissions are stored in the block algorithm and are determined dynamically in online mode.

### Permissions via the OS_Perm input parameter at the block itself

Controllable blocks have the OS_Perm input parameter which allows you to implement individual operator control strategies by setting the operator control permissions. A pressure relief valve, for example, can only be opened by the master control system. The operator may only close the valve. These authorizations are defined during configuration. These operator control permissions are displayed in the preview view of the faceplate. For information about setting the individual operator control permissions (OS_Perm) refer to the description of the functions of the individual blocks.

The relevant operator controls are enabled if the operator has suitable permissions. The block algorithm processes the input data.

**Note**

If you interconnect a parameter that is also listed in OS_Perm as a parameter, you have to reset the corresponding OS_Perm bit.

### See also

Enabling local operator authorization (Page 169)

## 2.3.4 Display of delay times

### Display of delay times

As soon as the immediate command output is delayed due to an active delay time at the "Large" blocks of the "Drives" family, this fact is signaled by a bit in the status word and a display in the faceplate.

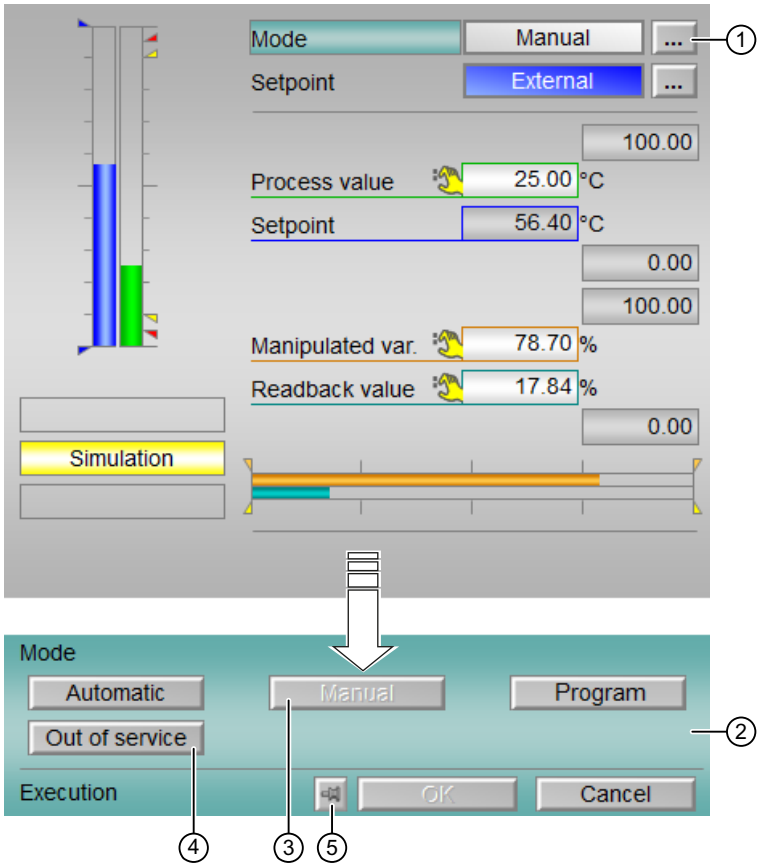
These times include:

- Pre-warning time
- On/Off delays
- Restart inhibit

## 2.3.5 Switching operating states and operating modes

### Requirements

You can change the operating state, operating mode and other parameters if needed in faceplates if you have the corresponding operator control permission (`OS_Perm`). This operator control permission can be configured in the engineering system (ES).




**(1)** The mouse cursor changes when you place it over the following button:




The mouse pointer now looks as follows:



When you click on the button with the mouse pointer, the bottom of the faceplate expands. You now see the field for changing the operating mode, for example.

**(2)** Field for changing the operating mode, operating state etc. This example describes changing the operating mode.

If the indicators for the operating state is currently located in this field and you have configured text for specific instances, this text is also shown. You can find more information about this in the section Labeling of buttons and text (Page 235).

**(3)** The text on this button is gray. You cannot select this operating mode due to the following reasons:

- This operator control permission for this operating mode cannot be configured in the engineering system (ES).

or

- The operating mode is already selected at this time.

or

- Due to the technology, you cannot switch from the operating mode currently set and the desired operating mode.

**(4)** The text on this button is black. You can switch to this operating mode.

### How to change the operating mode (using the PIDConL block in standard view as an example)

1. Click one of the selectable buttons in the operating mode field.
2. Confirm your selection by clicking "OK".
3. If you do not want to apply your selection, click "Cancel".

After clicking the "OK" or "Cancel" button, the faceplate is reduced again to its original form.

### (5) Multiple operation

If the operating window is not to close after the confirmation of a command, it can be "attached". The following button is located below the operating window for this purpose:

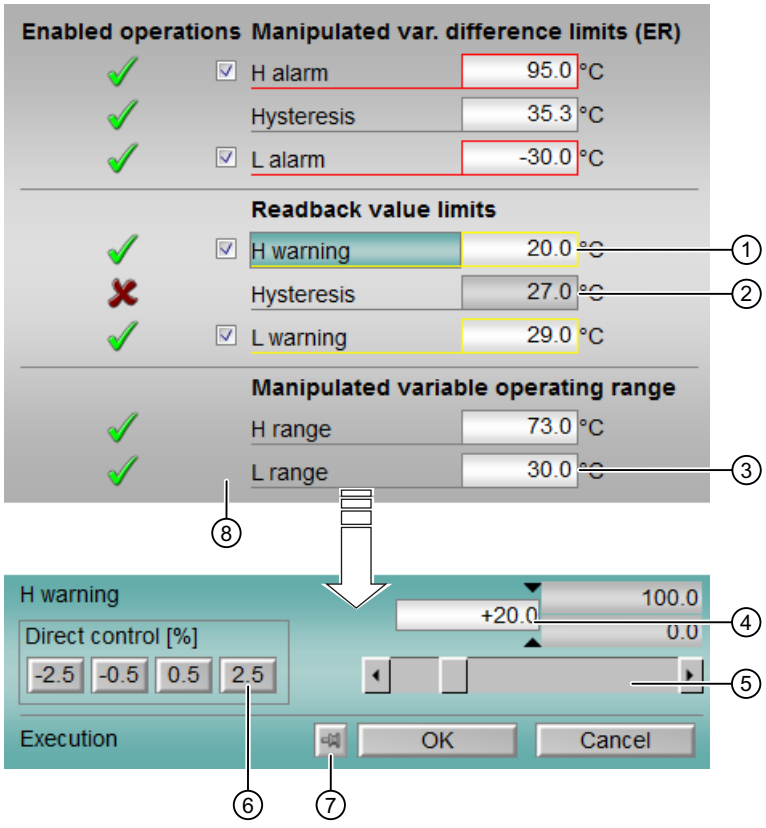

Operating window is closed after the value is applied


Operating window remains open after the value is applied

## 2.3.6 Changing values

### Requirements

You can change the values in faceplates and in the block icons if you have the corresponding operator authorization (`OS_Perm`). This operator permission can be configured in the engineering system (ES). The following example shows how values are changed via the faceplate.




**(1)** The background color of the input box is white. You can change the value. The mouse pointer changes when you place it over the input box:



**(2)** The background color of the input box is gray. You cannot change the value.

**(3)** If you click on the input box, the bottom of the faceplate expands. You now see the field for changing values.

### How to change values in faceplates

You have three options for changing values:

- "Direct control": Click on a button such as "2.5" in the box **(6)**. The value is immediately changed and applied. It is no longer necessary to confirm this value.
- Changing values using the slide control **(5)**: move the slide control until the desired value is shown in the box. Then confirm the value using the Enter key or by clicking "OK". Read the section "Setting the multiple step operation".
- Change the values in the input box **(4)**: Click the input box and enter the new value. If the new values are outside the limits, they are rejected and the old values are retained. Then confirm the value using the Enter key and clicking "OK". Read the section "Setting the multiple step operation".

### Setting the multiple step operation

You can use the internal @APLCommandExecutionSteps tag in the Tag Management of the WinCC Explorer to specify if values are to be changed in two or three steps.

Follow the steps outlined below:

1. Double-click on the internal @APLCommandExecutionSteps tag
2. Change the start value to 2 or 3 in the Limits/Reporting tab.
   **Start value = 2**: It is no longer necessary to confirm the value in the faceplate by clicking "OK"; values are applied immediately.
   **Start value = 3**: Each value change in the faceplate (with the exception of those for direct control) needs to be confirmed with "OK".

### Changing the values for "Direct control"

Specify the percentage values for the two inner keys for direct control with the `DirectOperationValue` property at the block icon. The outer two keys are automatically determined with `DirectOperationValue` times the factor 5.

If `DirectOperationValue` is not an integer but the values in the faceplate are integers, then `DirectOperationValue` is rounded up to the next integer.

If the rounded value is 0, 1 is used for `DirectOperationValue`.

The default value for `DirectOperationValue` is 0.5. With integer format, the "+/-1%" results for the inner keys and "+/-5%" for the outer keys.

The value change also depends on the change in the range of value, which is given by a high limit and a low limit:

| Blocks | Kind of value | High limit | Low limit |
|---|---|---|---|
| MotSpdCL | Setpoint | SP_HiLim | SP_LoLim |
| VlvAnL, VlvPosL | Manipulated variable MV | MV_HiLim | MV_LoLim |
| PIDConL, PIDConS, PIDConR, PIDStepL, FmCont, FmTemp | Setpoint,<br>Manual value Man | SP_HiLim,<br>ManHiLim | SP_LoLim,<br>ManLoLim |
| OpAnL | Setpoint | SP_HiLim | SP_LoLim |

**Note**

When the interlock is active, the High and Low limits of the operation area are considered for VlvPosL and VlvAnL blocks.

The value change is calculated as follow:

value change = button value * (high limit - low limit) / 100.0

**Note**

Analog value format should be configured to accommodate the smallest change triggered by operating range and `DirectOperationValue`. This ensures proper operation and no disabling of direct operation in operation area.

- Example 1:
  Operating range: 0 ~ 1
  Direct Operation Value: 0.5%
  Smallest possible change is 0.005.
  Hence AnalogValueFormat should have atleast 3 decimal places - ##0.000.
- Example 2:
  Operating range: 0 ~ 100
  Direct Operation Value: 0.5%
  Smallest possible change is 0.5.
  Hence AnalogValueFormat should have atleast 1 decimal place - ##0.0Properties

`DirectOperationValue`: Block Icon>Object Properties>Configurations>`DirectOperationValue`

`AnalogValueFormat`: Block Icon>Object Properties>Configurations>`AnalogValueFormat`

### (7) Multiple operation

If the operating window is not to close after the confirmation of a command, it can be "pinned". The following button is located below the operating window for this purpose:



Operating window unpinned.

When the value of a parameter is set using the "Direct control" button, the value is applied to the parameter and the "Cancel" button is disabled. Click "OK" to close the operating window.



Operating window pinned.

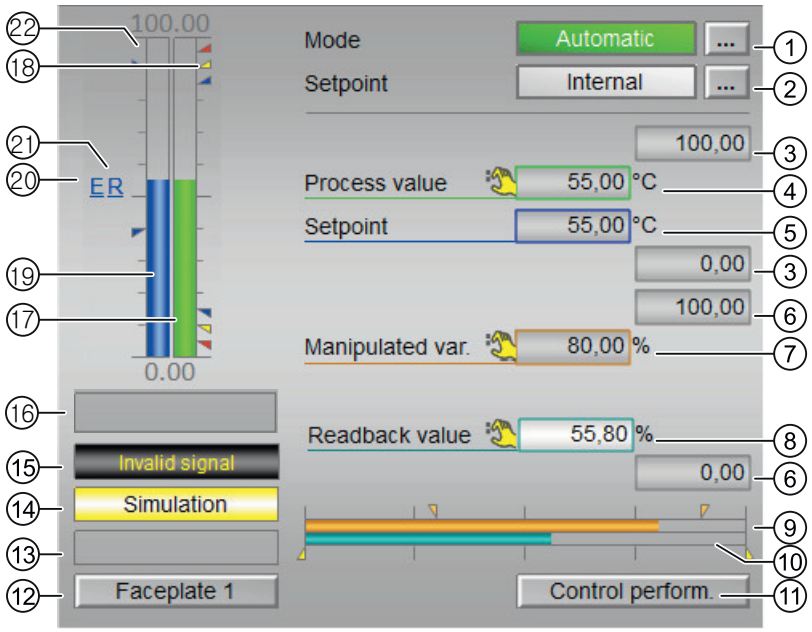
When the value of a parameter is applied using the "Direct control" button, the value is applied to the parameter and the "OK" button is disabled. Click "Cancel" to close the operating window.

### (8) Suppress messages

You can enable / disable messages by setting the check mark.

## 2.3.7 FM controllers standard view (analog)

### Standard view (analog) of FM controllers




### (1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 79)
- Automatic mode (Page 79)
- Program mode for controllers (Page 85)
- Out of service (Page 77)

Refer to chapter Switching operating states and operating modes (Page 289) for information on switching the operating mode.

### (2) Display and switch the setpoint specification

This area shows how to specify the setpoint. The setpoint can be specified as follows:

- By the application ("External", CFC/SFC)
- By the user directly in the faceplate ("Internal").

Refer to the Switching operating states and operating modes (Page 289) chapter for information on switching the setpoint specification.

For more information on the setpoint specification, refer to the Setpoint specification - internal/ external (Page 136) chapter.

If text is configure for these commands, it is displayed as additional text and as button labels for command selection. You can find more information about this in chapter Labeling of buttons and text (Page 235)

### (3) High and low scale range for the process value

These values provide information on the display range (PV_OpScale) for the bar graph of the process value. The scale range is defined in the Engineering System.

### (4) Display of the process value including signal status

This area provides information on the current process value (PV) with the corresponding signal status.

### (5) Display and change the setpoint including signal status

This area provides information on the current setpoint (SP) with the corresponding signal status.

Refer to the Changing values (Page 291) chapter for information on changing the setpoint. The setpoint specification **(2)** also needs to be set to "Internal" for this block.

### (6) High and low scale range for the setpoint

This area is already set and cannot be changed.

### (7) Display and change the manipulated variable including signal status

This area shows the current "Manipulated variable" (MV) with the corresponding signal status.

Refer to chapter Changing values (Page 291) for information on changing the manipulated variable. You can only make a change in manual mode.

### (8) Display of the position feedback including signal status

This display is only visible when the corresponding block input is connected.

This area provides information on the current readback value of the manipulated variable with the corresponding signal status. This display is only available when the readback value in the box is interconnected to the Rbk input parameter.

### (9) Bar graph for the "Manipulated variable"

This area shows the current "Manipulated variable" in the form of a bar graph (MV_OpScale). The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (10) Bar graph for position feedback

This display is only visible when the corresponding block input is connected.

This area shows the current position feedback in the form of a bar graph. The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (11) Navigation button for switching to the standard view of the ConPerMon block

Use this navigation button to reach the standard view of the ConPerMon block. The visibility of this navigation button depends on the configuration in the Engineering System (ES).

Refer also to chapter Opening additional faceplates (Page 234) for more on this.

### (12) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the Engineering System. The visibility of this navigation button depends on the configuration in the Engineering System (ES).

Refer also to chapter Opening additional faceplates (Page 234) for more on this.

### (13) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this is available in chapter Release for maintenance (Page 70) Display area for block states.

### (14) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"

You can find additional information on this in chapter Simulating signals (Page 64).

### (15) Display area for block states

This area provides additional information on the operating state of the block:

- "Internal error"
- "Invalid Signal"

### (16) Display area for block states

This area provides additional information on the operating state of the block (from high to low according to priority):

- "Fuzzy Optim." (FmCont only)
- "Tracking FB"
- "Tracking FM"

- "Safety mode FM"
- "Fuzzy control" (FmCont only)

**Note**

"Ramp active" indication is not available for FMCont and FMTemp blocks in the Standard view.

### (17) Limit display

These colored triangles indicate the configured limits in the respective bar graph:

- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

**Note**

The symbols displayed are not valid for user-configured message classes. Please take into consideration the validity of terms for User-configured message classes (Page 45).

### (18) Bar graph for the "Process value"

This area shows the current "Process value" in the form of a bar graph. The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (19) Bar graph for the "Setpoint"

This area shows the current "Setpoint" in the form of a bar graph. The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (20) Display of external setpoint

This display [E] is only visible when you have selected "Internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

### (21) Display for the target setpoint of the setpoint ramp

This display [R] shows the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 334).

### (22) Limit display for the setpoint

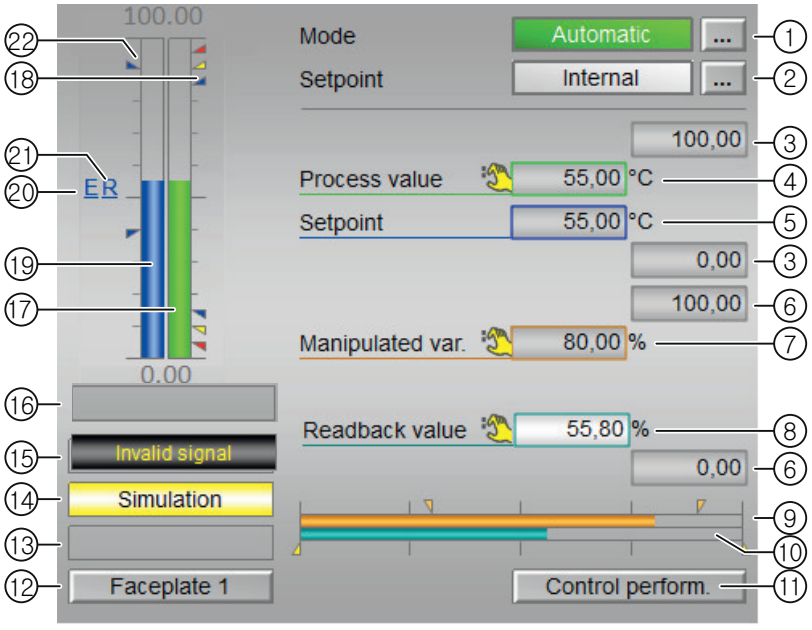
These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the ES.

### (23) High and low scale range of bar graph

These values provide the high and low limit values of the bargraph.

## 2.3.8 FM controllers standard view (pulse controller)

### Standard view (pulse) of FM controllers




### (1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 79)
- Automatic mode (Page 79)
- Program mode for controllers (Page 85)
- Out of service (Page 77)

Refer to chapter Switching operating states and operating modes (Page 289) for information on switching the operating mode.

### (2) Display and switch the setpoint specification

This area shows how to specify the setpoint. The setpoint can be specified as follows:

- By the application ("External", CFC/SFC)
- By the user directly in the faceplate ("Internal").

Refer to the Switching operating states and operating modes (Page 289) chapter for information on switching the setpoint specification.

For more information on the setpoint specification, refer to the Setpoint specification - internal/ external (Page 136) chapter.

### (3) High and low scale range for the process value

These values provide information on the display range (`PV_OpScale`) for the bar graph of the process value. The scale range is defined in the Engineering System.

### (4) Display of the process value including signal status

This area provides information on the current process value (`PV`) with the corresponding signal status.

### (5) Display and change the setpoint including signal status

This area provides information on the current setpoint (`SP`) with the corresponding signal status.

Refer to the Changing values (Page 291) chapter for information on changing the setpoint. The setpoint specification **(2)** also needs to be set to "Internal" for this block.

### (6) High and low scale range for the setpoint

This area is already set and cannot be changed.

### (7) Display and change the manipulated variable including signal status

This area shows the current "Manipulated variable" (`MV`) with the corresponding signal status.

Refer to chapter Changing values (Page 291) for information on changing the manipulated variable. You can only make a change in manual mode.

### (8) Display of the position feedback including signal status

This display is only visible when the corresponding block input is interconnected.

This area shows the current feedback of the manipulated variable with the corresponding signal status. This display is only available when the readback value in the box is interconnected to the `Rbk` input parameter.

### (9) Bar graph for the "Manipulated variable"

This area shows the current "Manipulated variable" in the form of a bar graph (`MV_OpScale`). The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (10) Bar graph for position feedback

This display is only visible when the corresponding block input is interconnected.

This area shows the current position feedback in the form of a bar graph. The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (11) Button for switching to the standard view of the ConPerMon block

Use this button for the standard view of the ConPerMon block. The visibility of this button depends on the configuration in the Engineering System (ES).

Refer also to chapter Opening additional faceplates (Page 234) for more on this.

### (12) Button for switching to the standard view of any faceplate

Use this button for the standard view of a block configured in the engineering system. The visibility of this button depends on the configuration in the Engineering System (ES).

Refer also to chapter Opening additional faceplates (Page 234) for more on this.

### (13) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this is available in chapter Release for maintenance (Page 70) Display area for block states.

### (14) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"

You can find additional information on this in section Simulating signals (Page 64).

### (15) Display area for block states

This area provides additional information on the operating state of the block:

- "Internal error"
- "Invalid Signal"

### (16) Display area for block states

This area provides additional information on the operating state of the block (from high to low according to priority):

- "Fuzzy Optim." (FmCont only)
- "Tracking FB"
- "Tracking FM"
- "Safety mode FM"
- "Fuzzy control" (FmCont only)

**Note**

"Ramp active" indication is not available for FMCont and FMTemp blocks in the Standard view.

### (17) Bar graph for the "Process value"

This area shows the current "Process value" in the form of a bar graph. The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (18) Limit display

These colored triangles indicate the configured limits in the respective bar graph:

- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

**Note**

The symbols displayed are not valid for user-configured message classes. Please take into consideration the validity of terms for User-configured message classes (Page 45).

### (19) Bar graph for the "Setpoint"

This area shows the current "Setpoint" in the form of a bar graph. The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (20) Display of external setpoint

This display [E] is only visible when you have selected "Internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

### (21) Display for the target setpoint of the setpoint ramp

This display [R] shows the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 334).

### (22) Limit display for the setpoint

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the ES.

### (23) High and low scale range of bar graph

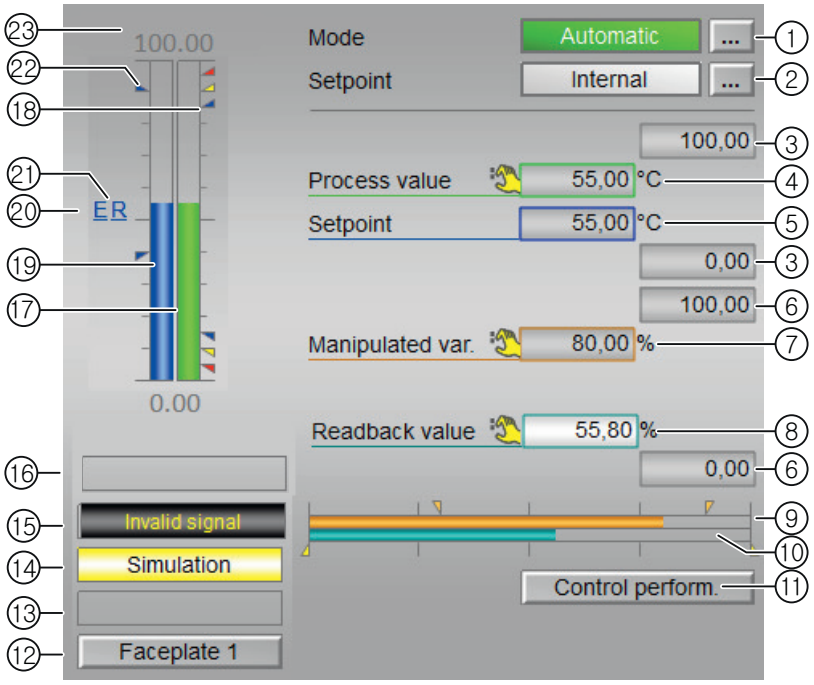
These values provide the high and low limit values of the bargraph.

### See also

Labeling of buttons and text (Page 235)

## 2.3.9 FM controllers standard view (step controller with position feedback)

### Standard view with position feedback of FM controllers




### (1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 79)
- Automatic mode (Page 79)
- Program mode for controllers (Page 85)
- Out of service (Page 77)

Refer to chapter Switching operating states and operating modes (Page 289) for information on switching the operating mode.

### (2) Display and switch the setpoint specification

This area shows how to specify the setpoint. The setpoint can be specified as follows:

- By the application ("External", CFC/SFC)
- By the user directly in the faceplate ("Internal").

Refer to the Switching operating states and operating modes (Page 289) chapter for information on switching the setpoint specification.

For more information on the setpoint specification, refer to the Setpoint specification - internal/ external (Page 136) chapter.

### (3) High and low scale range for the process value

These values provide information on the display range (`PV_OpScale`) for the bar graph of the process value. The scale range is defined in the Engineering System.

### (4) Display of the process value including signal status

This area provides information on the current process value (`PV`) with the corresponding signal status.

### (5) Display and change the setpoint including signal status

This area provides information on the current setpoint (`SP`) with the corresponding signal status.

Refer to the Changing values (Page 291) chapter for information on changing the setpoint. The setpoint specification **(2)** also needs to be set to "Internal" for this block.

### (6) High and low scale range for the setpoint

This area is already set and cannot be changed.

### (7) Display and change the manipulated variable including signal status

This area shows the current "Manipulated variable" (`MV`) with the corresponding signal status.

Refer to chapter Changing values (Page 291) for information on changing the manipulated variable. You can only make a change in manual mode.

### (8) Display of the position feedback including signal status

This display is only visible when the corresponding block input is interconnected.

This area shows the current feedback of the manipulated variable with the corresponding signal status. This display is only available when the readback value in the box is interconnected to the `Rbk` input parameter.

### (9) Bar graph for the "Manipulated variable"

This area shows the current "Manipulated variable" in the form of a bar graph (`MV_OpScale`). The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (10) Bar graph for position feedback

This display is only visible when the corresponding block input is interconnected.

This area shows the current position feedback in the form of a bar graph. The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (11) Button for switching to the standard view of the ConPerMon block

Use this button for the standard view of the ConPerMon block. The visibility of this button depends on the configuration in the Engineering System (ES).

Refer also to chapter Opening additional faceplates (Page 234) for more on this.

### (12) Button for switching to the standard view of any faceplate

Use this button for the standard view of a block configured in the Engineering System. The visibility of this button depends on the configuration in the Engineering System (ES).

Refer also to chapter Opening additional faceplates (Page 234) for more on this.

### (13) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this is available in chapter Release for maintenance (Page 70) Display area for block states.

### (14) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"

You can find additional information on this in section Simulating signals (Page 64).

### (15) Display area for block states

This area provides additional information on the operating state of the block:

- "Internal error"
- "Invalid Signal"

### (16) Display area for block states

This area provides additional information on the operating state of the block (from high to low according to priority):

- "Fuzzy Optim." (FmCont only)
- "Tracking FB"
- "Tracking FM"
- "Safety mode FM"
- "Fuzzy control" (FmCont only)

**Note**

"Ramp active" indication is not available for FMCont and FMTemp blocks in the Standard view.

### (17) Bar graph for the "Process value"

This area shows the current "Process value" in the form of a bar graph. The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (18) Limit display

These colored triangles indicate the configured limits in the respective bar graph:

- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

**Note**

The symbols displayed are not valid for user-configured message classes. Please take into consideration the validity of terms for User-configured message classes (Page 45).

### (19) Bar graph for the "Setpoint"

This area shows the current "Setpoint" in the form of a bar graph. The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (20) Display of external setpoint

This display [E] is only visible when you have selected "Internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

### (21) Display for the target setpoint of the setpoint ramp

This display [R] shows the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 334).

### (22) Limit display for the setpoint

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the ES.

### (23) High and low scale range of bar graph

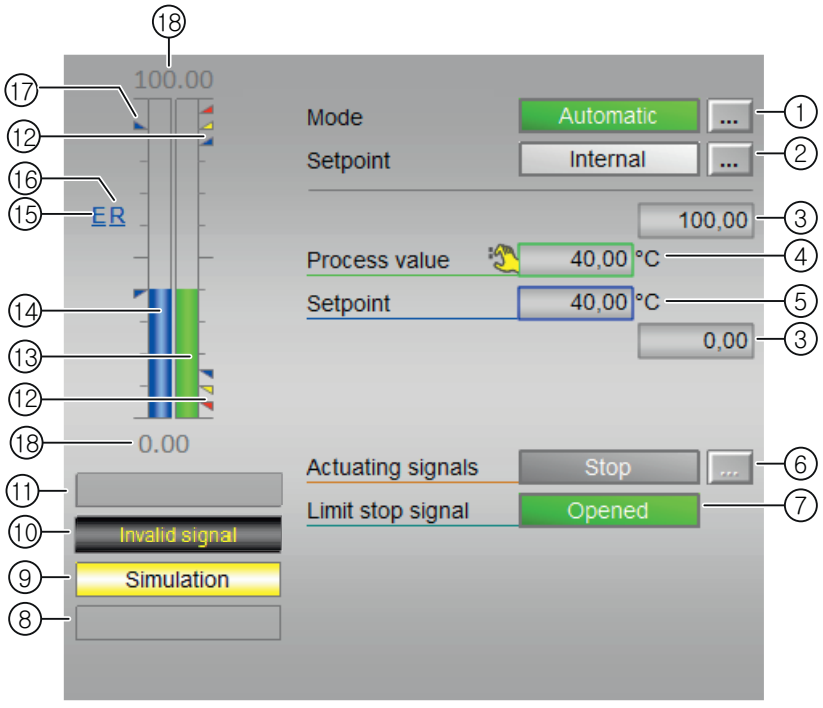
These values provide the high and low limit values of the bargraph.

### See also

Labeling of buttons and text (Page 235)

## 2.3.10 FM controllers standard view (step controller without position feedback)

### Standard view without position feedback of FM controllers



### (1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 79)
- Automatic mode (Page 79)
- Program mode for controllers (Page 85)
- Out of service (Page 77)

Refer to chapter Switching operating states and operating modes (Page 289) for information on switching the operating mode.

### (2) Display and switch the setpoint specification

This area shows how to specify the setpoint. The setpoint can be specified as follows:

- By the application ("External", CFC/SFC)
- By the user directly in the faceplate ("Internal").

Refer to the Switching operating states and operating modes (Page 289) chapter for information on switching the setpoint specification.

For more information on the setpoint specification, refer to the Setpoint specification - internal/ external (Page 136) chapter.

### (3) High and low scale range for the process value

These values provide information on the display range (`PV_OpScale`) for the bar graph of the process value. The scale range is defined in the Engineering System.

### (4) Display of the process value including signal status

This area provides information on the current process value (`PV`) with the corresponding signal status.

### (5) Display and change the setpoint including signal status

This area provides information on the current setpoint (`SP`) with the corresponding signal status.

Refer to the Changing values (Page 291) chapter for information on changing the setpoint. The setpoint specification **(2)** also needs to be set to "Internal" for this block.

### (6) Operating and displaying the actuating signal

This area shows the current feedback of the actuating signal.

- "Open"
- "Stop"
- "Close"

The button is shown next to the display in manual mode. You can influence the actuating signal here. Refer to the Switching operating states and operating modes (Page 289) chapter for more on this.

If text is configured for these commands, it is displayed as status text and as button labels for command selection. You can find more information about this in section Labeling of buttons and text (Page 235)

### (7) Display of the limit stop value including signal status

This area shows the limit stop signal with the corresponding signal status.

- "Open"
- "Closed"

If text is configured for these commands, it is displayed as status text and as button labels for command selection. You can find more information about this in section Labeling of buttons and text (Page 235)

### (8) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this is available in chapter Release for maintenance (Page 70) Display area for block states.

### (9) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"

You can find additional information on this in section Simulating signals (Page 64).

### (10) Display area for block states

This area provides additional information on the operating state of the block:

- "Internal error"
- "Invalid Signal"

### (11) Display area for block states

This area provides additional information on the operating state of the block (from high to low according to priority):

- "Fuzzy Optim." (FmCont only)
- "Tracking FB"
- "Tracking FM"

- "Safety mode FM"
- "Fuzzy control" (FmCont only)

**Note**

"Ramp active" indication is not available for FMCont and FMTemp blocks in the Standard view.

### (12) Limit display

These colored triangles indicate the configured limits in the respective bar graph:

- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

**Note**

The symbols displayed are not valid for user-configured message classes. Please take into consideration the validity of terms for User-configured message classes (Page 45).

### (13) Bar graph for the "Process value"

This area shows the current "Process value" in the form of a bar graph. The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (14) Bar graph for the "Setpoint"

This area shows the current "Setpoint" in the form of a bar graph. The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (15) Display of external setpoint

This display [E] is only visible when you have selected "Internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

### (16) Display for the target setpoint of the setpoint ramp

This display [R] shows the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 334).

### (17) Limit display for the setpoint

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the ES.

### (18) High and low scale range of bar graph

These values provide the high and low limit values of the bargraph.

### See also

Opening additional faceplates (Page 234)

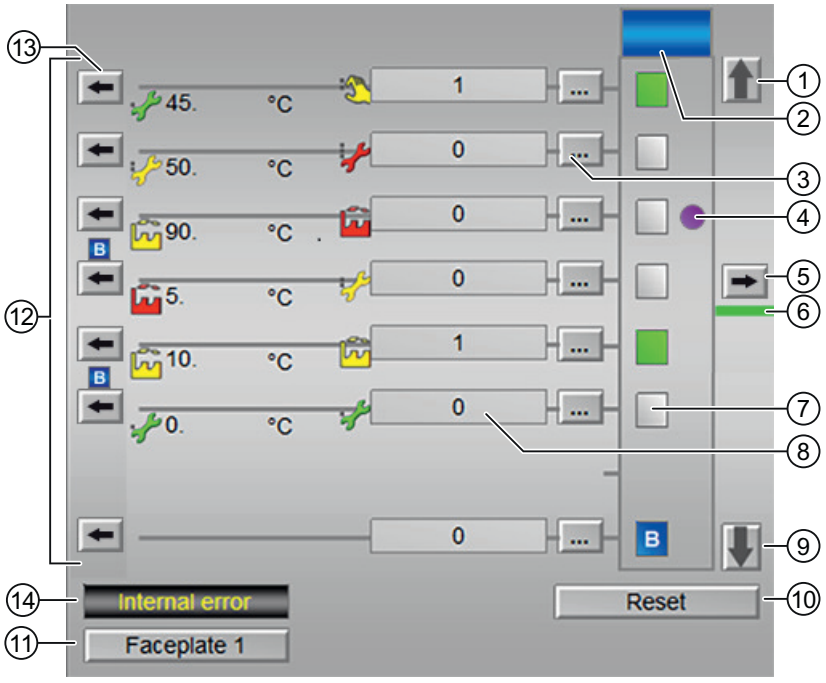
## 2.3.11 Interlock blocks standard view

### Interlock blocks standard view Intlk02, Intlk04, Intlk08, Intlk16

The number of the displayed input values depends on the interlock block you have selected.

The operation and functions are identical for all interlock blocks and do not depend on the number of input values.

The **Intlk16** interlock block has two additional buttons for switching between the input values 1 to 8 and 9 to 16.




### (1), (9) Switching between the input values 1 to 8 and 9 to 16 (for Intlk16 only)

The buttons (1) or (9) are displayed depending on the view you are in. These buttons are only available for the **Intlk16** block.

The **Intlk16** block provides two views:

- When you are in the first view, the input values 1 to 8 are available in the area (12). The button (9) is displayed. You switch to the second view by clicking on the button (9).
- When you are in the second view, the input values 9 to 16 are available in the area (12). The button (1) is displayed. You switch back to the first view by clicking on the button (1).

### (2) Status of the output signal of the interlock block

This area (2) shows the status of the output signal of the interlock block (priority from high to low). You can configure the logic in the engineering system (ES).

| | Logic | |
|---|---|---|
| **Color of the field** | **AND** | **OR** |
| Gray | Block is not used (only for display in the faceplate, interlock logic still enabled), set using the `NotUsed` = 1 parameter | |
| Blue | Excluded (bypass) | |
| Yellow | Simulated | |
| Red | Interlocked | |
| Green | Not interlocked | |

### (3) Exclude input values

You can use the button (3) to exclude input values from processing. Depending on the previous settings, you can "Set" or "Reset" this property.

If the input value has been excluded, the following symbol appears in the field (8):




For more information on the operation, refer to the section Switching operating states and operating modes (Page 289).

**Note**

This function can only be executed in the faceplate with "high-level operating permission".

**Note**

Operator permissions via `OS_Perm` do not depend on the `Feature.Bit5` setting. You can find additional information in section Activate OS_Perm bits (Page 168).

### (4) "First in" status display

The following symbol is displayed next to an input value, if this input value has caused the last output signal change from 1 to 0 (good state to locked):



You can reset the first-in (initial) signal with the button (10).

**Note**

This function can only be executed in the faceplate with "process control" operating permission.

You can find additional information on this in the section Recording the first signal for interlock blocks (Page 58).

For more information on the operation, refer to the section Switching operating states and operating modes (Page 289).

### (5) Open faceplate of the output value

When you press the button (5), you can open the faceplate associated with the output value. The function of this button depends on the configuration in the engineering system (ES). See also the section Opening additional faceplates (Page 234) for more on this.

### (6) Status of the block output

The line color indicates the status of the block output:

| Color of the line | Output status |
|---|---|
| Green | Output is enabled |
| White | Output is disabled |

### (7) Display the status for further processing

The symbol shows the status for further processing of the input values:

| Icon | Further processing |
|---|---|
|  | The input value is processed further with value 1. |
|  | The input value is processed further with value 0. |
|  | The input value is excluded from further processing. |

### (8) Display of input values (BOOL) with signal status (in front of the field)

These fields show the interlock information associated with the analog value (13) with a signal status:

- 1 = "Good" state
- 0 = "Locked"

**Changing the display**

You can change the displays for 0 and 1 in the CFC in the object properties of the Interlock block:

- Navigate to the I/Os (object properties).
- Change the default for the input parameters (`Inxx`) in the Text 0 and Text 1 columns to what you later want to see in runtime.

### (9) Switching input values

Read point (1) for this.

### (10) "Reset" the settings for further processing

When you press the button (10), you can "Reset" all input values:

- "Reset exclusions": the exclusions of the input values are reset.
- "Reset first-in": First-in detection / status display (4) is reset.

You can find additional information on this in the section Recording the first signal for interlock blocks (Page 58).

### (11) Navigation button for switching to the standard view of any faceplate

Use this navigation button to open the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234).

### (12) Displaying analog input values

The interconnected analog input values (`AVxx`) are displayed in this area. Set a unit of measure (`AVxx_Unit` as shown in the picture with [Unit]) for each input in the engineering system (ES).

**Changing the display**

You can change the displays in the CFC in the object properties of the Interlock block:

- Navigate to the I/Os (object properties).
- In the “OS additional text” column, change the default setting for the input parameter (`INxx`) to what you want to see during runtime later.
- In the "Identifier" column, change the default setting for the input parameter (`AVxx`) to what you want to see during runtime later.

- The text is used as a label and is therefore always displayed, in other words, it is independent of the signal status of the corresponding AVxxx input.
- The font size is reduced during the runtime if the input text length is greater than the label width.

**Note**

If a text is added to the "OS additional text" field of the input parameter (INxx), this text will be displayed even if the "Identifier" text field of the input parameter (AVxx) is not empty. If the "OS additional text" field of the input parameter (INxx) is empty, the "Identifier" text of the input parameter (AVxx) is displayed.

The number of input values may vary depending on the selected interlock block:

- **Intlk02:** the input values 1 and 2 are available.
- **Intlk04:** the input values 1 to 4 are available.
- **Intlk08:** the input values 1 to 8 are available.
- **Intlk16:** the input values 1 to 8 are available. The input values 9 to 16 become available by pressing the button (9). You can find additional information on this topic in the description for (1) and (9).

### (13) Open faceplate of the input value

When you press the button (13), you can open the faceplate associated with each input value. The function of this button depends on the configuration in the engineering system (ES). See also the section Opening additional faceplates (Page 234) for more on this.

The connected block is in bypass if the following small bypass symbol is displayed over the arrow symbol:




This display can be used only if Feature2.Bit2 Separate evaluation for excluded and simulated interlock signals (Page 162) is active. If additionally the input BypEn.Inx = 1, this interlock input will be set in bypass.

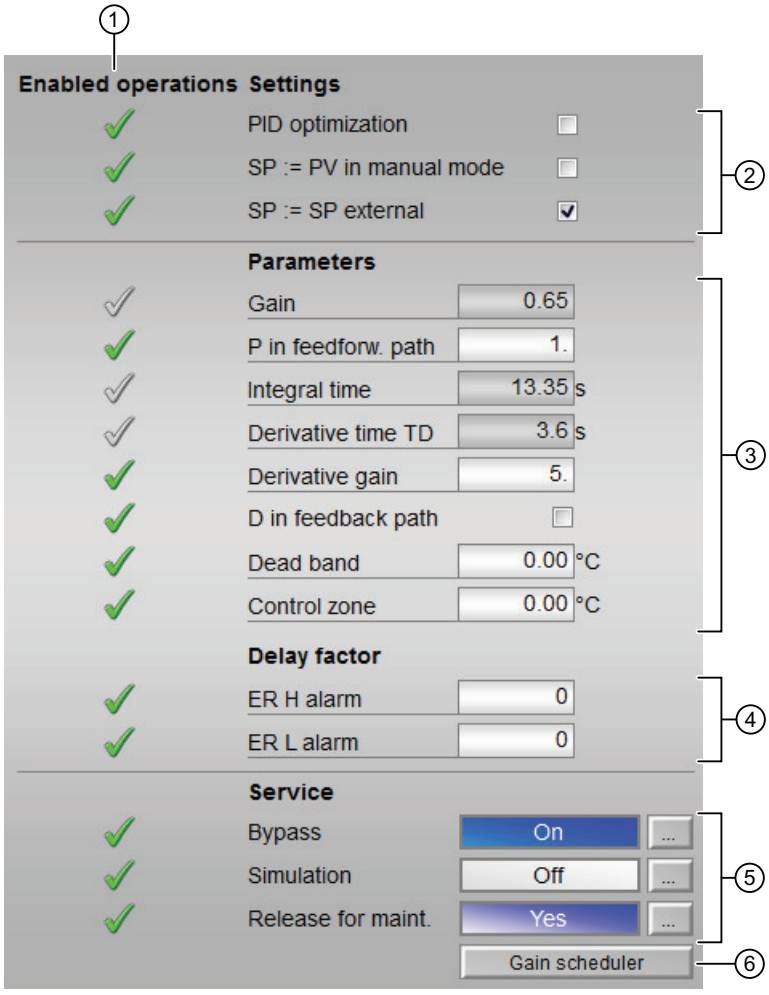
### (14) Display area for block states

This area provides additional information on the operating state of the block:

- "Internal error"

## 2.3.12 Parameter view of PID controllers

### Parameter view of PID controllers




### (1) "Enabled operation"

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

### (2) "Settings"

You can activate the following functions for the controller in this area:

- "PID optimization": ☑ (OptimEn) With pushing the button “Start Controller Optimization” in the tool “PCS7 PID-Tuner” a controller optimization will be activated.
- "SP := PV in manual mode": ☑ Bumpless switchover from manual mode to automatic mode
- "SP := SP external": ☑ Bumpless switchover of the setpoint for setpoint switchover from "external" to "internal". The internal setpoint is tracked to the external one.
  - With the PIDConR block, this area is only visible if you have set the `Feature` bit Switching operator controls for external setpoint to visible (Page 154) to 1.

### (3) "Parameters"

In this area, you change parameters and therefore influence the controller. Refer to the Changing values (Page 291) section for more on this.

You can influence the following parameters:

- "Gain": Proportional gain
- "P in feedforw path" : Proportional action to the feedback path (0 to 1),
  - 0 = Proportional action is completely in the feedback path
  - 1 = Proportional action is completely in the feedforward path

  (only with PIDConL, PIDConR, and PIDStepL)
- "Integral time": Integral action time in [s]
- "Derivative time TD": Derivative action time in [s]
- "Derivative gain": Gain of the derivative action
- "D in feedback path": Derivative action is moved to the feedback path (only with PIDConL, PIDConR, and PIDStepL)
- "Dead band": Width of dead band
   Dead band is temporarily disabled
- "Control zone": Width of the control zone (only with PIDConL block)
- "Motor actuating time": Motor actuating time [s] (for PIDStepL block only)
- "Minimum pulse duration": Minimum pulse duration [s] (for PIDStepL block only)
- "Minimum break duration": Minimum break duration [s] (for PIDStepL block only)

### (4) “Delay factor” (only for PIDConL and PIDConR)

In this area, you can change the following parameters:

- "ER H alarm": Delay factor at the positive setpoint step changes for incoming alarms at the control deviation monitoring `ER_AH_Lim`.
- "ER L alarm": Delay factor at the negative setpoint step changes for incoming alarms at the control deviation monitoring `ER_AL_Lim`.

### (5) "Service"

You can select the following functions in this area:

- "Bypass" (only with PIDConL, PIDConR, and PIDStepL)
- "Simulation"
- "Release for maintenance" (with display for a maintenance request)

Refer to the Switching operating states and operating modes (Page 289) section for more on this.

You can find information on this area in the section:

- Bypassing signals (Page 115) (for PIDConL, PIDConR, and PIDStepL)
- Simulating signals (Page 64)
- Release for maintenance (Page 70)

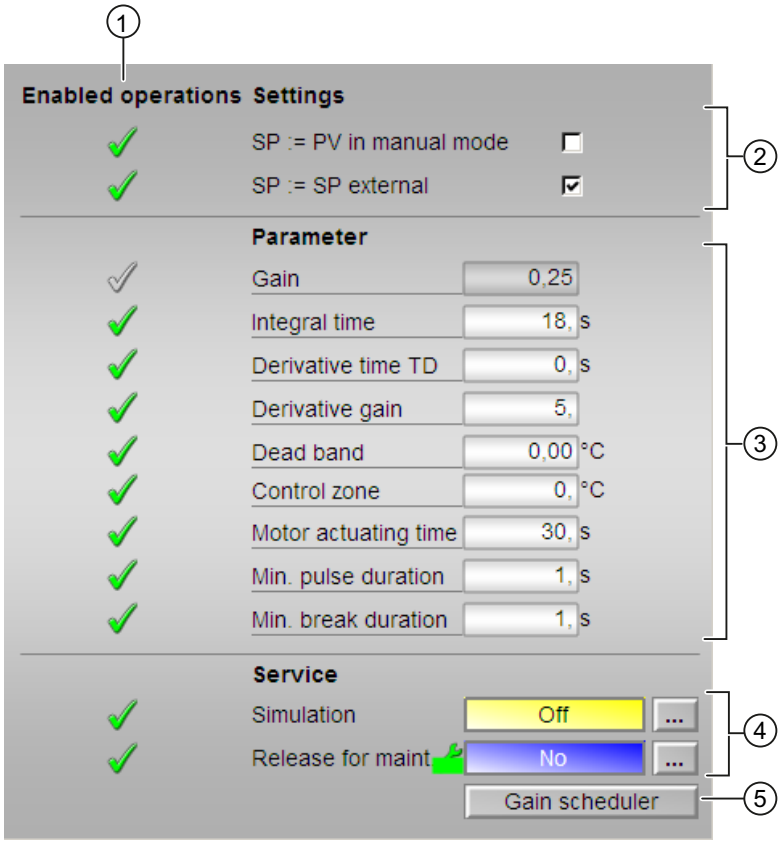
### (6) Navigation button for the GainSched block

You can use this navigation button to reach the GainSched block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234).

## 2.3.13 Parameter view of FM controllers

### Parameter view of FM controllers




### (1) "Enabled operations"

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

### (2) "Settings"

You can activate the following functions for the controller in this area:

- "SP := PV in manual mode": ☑ Bumpless switchover from manual mode to automatic mode
- "SP := SP external": ☑ Bumpless switchover of the setpoint for setpoint switchover from "external" to "internal" The internal setpoint is tracked to the external one.

### (3) "Parameters"

In this area, you change parameters and therefore influence the controller. Refer to the Changing values (Page 291) section for more on this.

You can influence the following parameters:

- "Gain": Proportional gain
- "Integral time" Integral action time in [s]
- "Derivative time TD": Derivative action time in [s]
- "Derivative gain": Gain of the derivative action
- "Dead band": Width of dead band
- "Control zone": Width of the control zone (for block FmTemp only)
- "Motor actuating time": Motor actuating time [s]
- "Minimum pulse duration": Minimum pulse duration [s]
- "Minimum break duration": Minimum break duration [s]

### (4) "Service"

You can activate the following functions in this area:

- "Simulation"
- "Release for maintenance" (with display for a maintenance request)

Refer to the Switching operating states and operating modes (Page 289) section for more on this.

You can find information on this area in the section:

- Simulating signals (Page 64)
- Release for maintenance (Page 70)

### (5) Navigation button for the GainSched block

You can use this navigation button to reach the GainSched block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

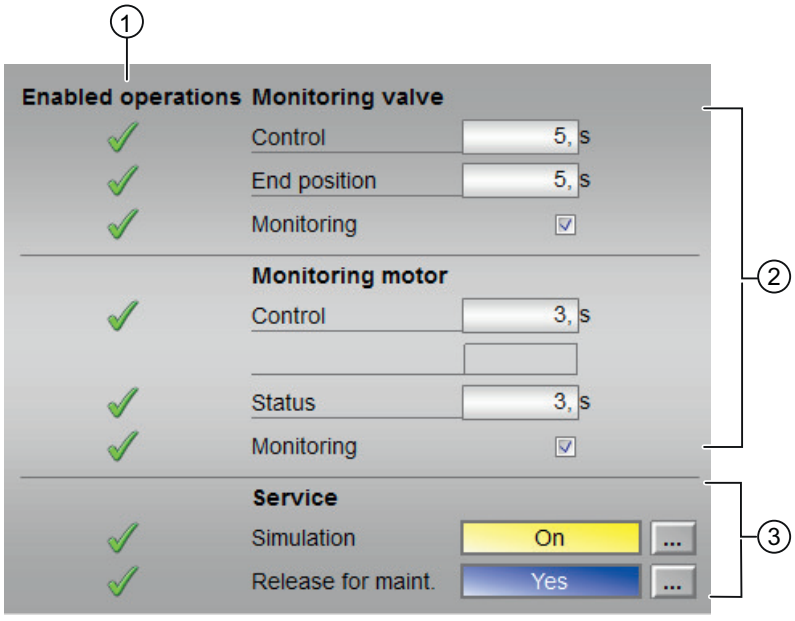
You can find additional information on this in the Opening additional faceplates (Page 234) section.

## 2.3.14 Parameter view for motors and valves

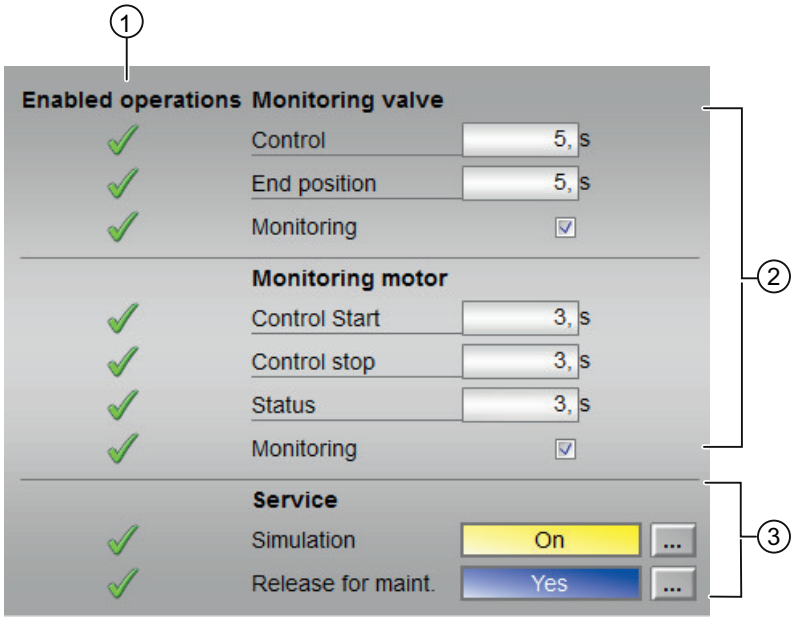
### Parameter view for motors and valves

The following parameter view applies to the following bocks:

- MotL - Motor (Large) (Page 1376)
- MotRevL - Reversible motor (Page 1440)
- MotS - Motor (Small) (Page 1415)
- MotSpdL - Two-speed motor (Page 1556)
- VlvL - Valve (Large) (Page 1691)
- VlvS - Valve (small) (Page 1728)




Parameter view for MotL, MotRevL, MotS, MotSpdL, VlvL and VlvS with `Feature.Bit13` = 0.




Parameter view for MotL, MotRevL, MotS, MotSpdL, VlvL and VlvS with `Feature.Bit13` = 1.

## (1) "Enabled operation"

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

## (2) "Monitoring"

In this area, you change parameters and therefore influence the motor. Refer to the Changing values (Page 291) section for more on this.

You can influence the following parameters:

- "Control": Monitoring time during startup and shutdown of the motor (dynamic)
- "Control stop": Monitoring time during shutdown of the motor (dynamic)
  only for MotL, MotRevL, MotSpdL and MotS with `Feature.Bit13` = 1
- "Control start": Monitoring time during startup of the motor (dynamic)
  only for MotL, MotRevL, MotSpdL and MotS with `Feature.Bit13` = 1
  "End position": Monitoring time during permanent operation of the motor (static)
- "Status": Monitoring time during permanent operation of the motor (static)
  The state is not displayed for small blocks.

**Enabling "Monitoring"**

You can enable monitoring by selecting the check box (☑)

You can find additional information on this in the section Monitoring the feedbacks (Page 104).

### (3) "Service"

You can select the following functions in this area:

- "Simulation"
- "Release for maintenance" (with display for a maintenance request)

Refer to the Switching operating states and operating modes (Page 289) section for more on this.

You can find information on this area in the section:

- Simulating signals (Page 64)
- Release for maintenance (Page 70)

### See also

Description of VlvS (Page 1728)

## 2.3.15 Limit view of FM controllers

### Limit view of FM controllers

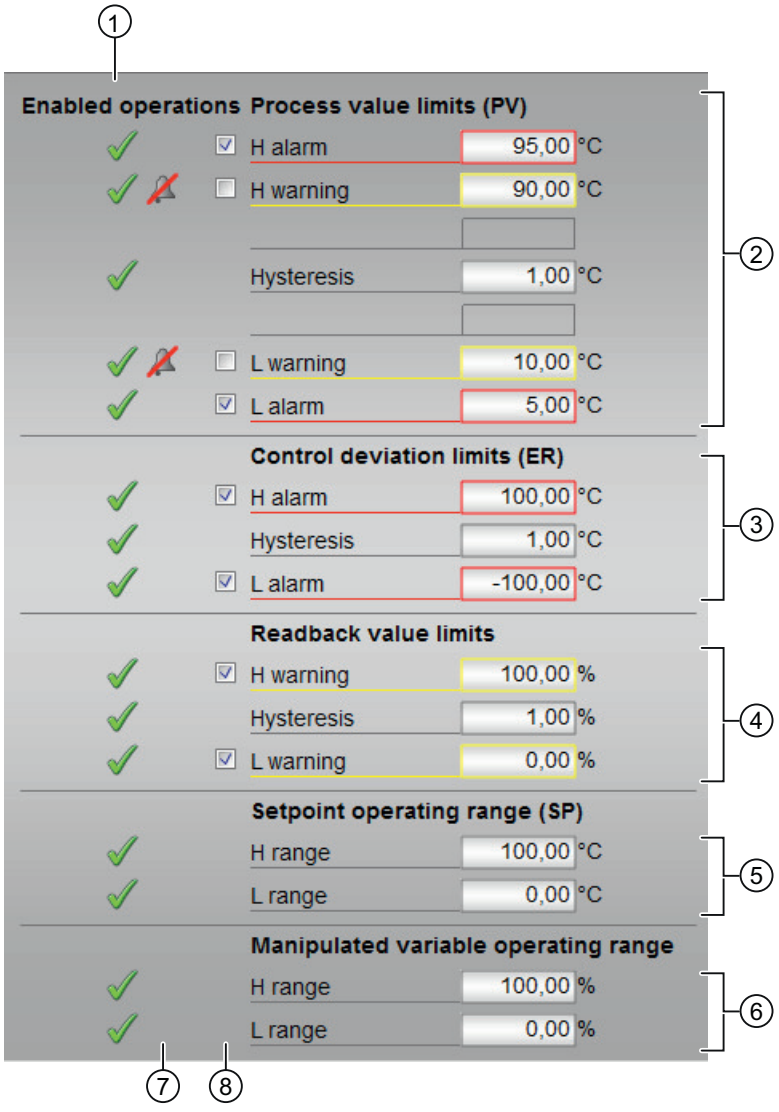
Several values are set in this view by default:

- Process value limits
- Error signal limits
- Readback value limits
- Setpoint operation range

The toolbars of the faceplate and the block icon indicate when the limits are reached or violated.

**Note**

The symbols displayed are not valid for user-configured message classes. Take into consideration the validity of terms for User-configured message classes (Page 45)




## (1) "Enabled operation"

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

### (2) "Process value limits (PV)"

In this area, you can enter the limits for the process value. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H alarm": Alarm high
- "H warning": Warning high
- "H tolerance": Tolerance high
- "Hysteresis"
- "L tolerance": Tolerance low
- "L warning": Warning low
- "L alarm": Alarm low

### (3) "Error signal limits (ER)"

In this area, you can enter the limits for the control deviation. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H alarm": Alarm high
- "Hysteresis"
- "L alarm": Alarm low

### (4) "Readback value limits (RBK)"

In this area, you can enter the limits for the readback value. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H warning": Warning high
- "Hysteresis"
- "L warning": Warning low

### (5) "Setpoint operation range (SP)"

In this area, you can enter the limits for the setpoint operation range. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H range": Range limit high
- "L range": Range limit low

### (6) "Manipulated variable operating range"

In this area, you can enter the limits for the manipulated variable operation range. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H range": Range limit high
- "L range": Range limit low

### (7) "Message suppression/delay"

Message suppression indicates whether or not the suppression of the associated message in the AS block is activated with the `xx_MsgEn` parameter. The output of messages is not suppressed when the block is installed (all `xx_MsgEn` parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

Alarm delays are also displayed in this position; for more on this see section Area of application of the alarm delays (Page 225).

### (8) Suppressing messages

You can enable / disable messages by setting the check mark.

## 2.3.16 Limit view of PID controllers

### Limit view of PID controllers

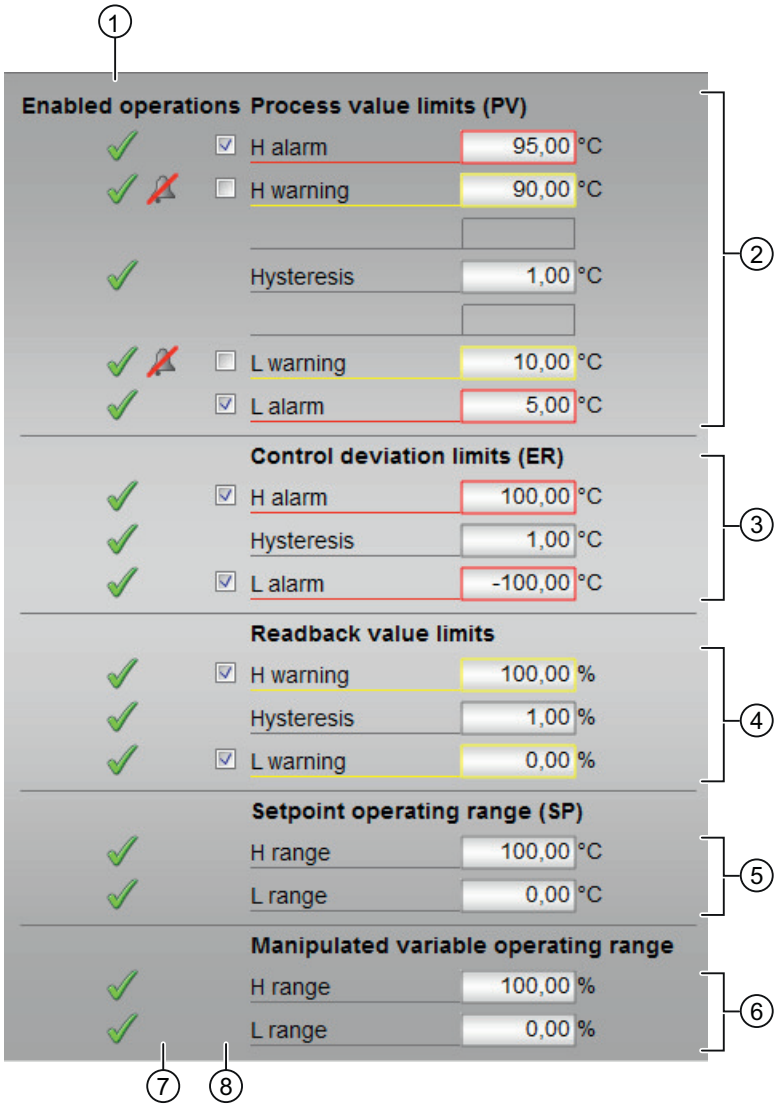
Several values are set in this view by default:

- Process value limits
- Error signal limits
- Readback value limits
- Setpoint operation range

The toolbars of the faceplate and the block icon indicate when the limits are reached or violated.

**Note**

The symbols displayed are not valid for user-configured message classes. Take into consideration the validity of terms for User-configured message classes (Page 45).




## (1) "Enabled operation"

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

### (2) "Process value limits (PV)"

In this area, you can enter the limits for the process value. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H alarm": Alarm high
- "H warning": Warning high
- "H tolerance": Tolerance high (not with PIDConS)
- "Hysteresis"
- "L tolerance": Tolerance low (not with PIDConS)
- "L warning": Warning low
- "L alarm": Alarm low

### (3) "Error signal limits (ER)" (not with PIDConS)

In this area, you can enter the limits for the control deviation. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H alarm": Alarm high
- "Hysteresis"
- "L alarm": Alarm low

### (4) "Readback value limits (RBK)" (not with PIDConS)

In this area, you can enter the limits for the readback value. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H warning": Warning high
- "Hysteresis"
- "L warning": Warning low

### (5) Setpoint operation range (SP)

In this area, you can enter the limits for the setpoint operation range. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H range": Range limit high
- "L range": Range limit low

### (6) Manipulated variable operating range

In this area, you can enter the limits for the manipulated variable operation range. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H range": Range limit high
- "L range": Range limit low

### (7) "Message suppression / delay"

Message suppression indicates whether or not the suppression of the associated message in the AS block is activated with the `xx_MsgEn` parameter. The output of messages is not suppressed when the block is installed (all `xx_MsgEn` parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

Alarm delays are also displayed in this position; for more on this see section Area of application of the alarm delays (Page 225).

### (8) Suppress messages

You can enable / disable messages by setting the check mark.

## 2.3.17 Limit view of motors

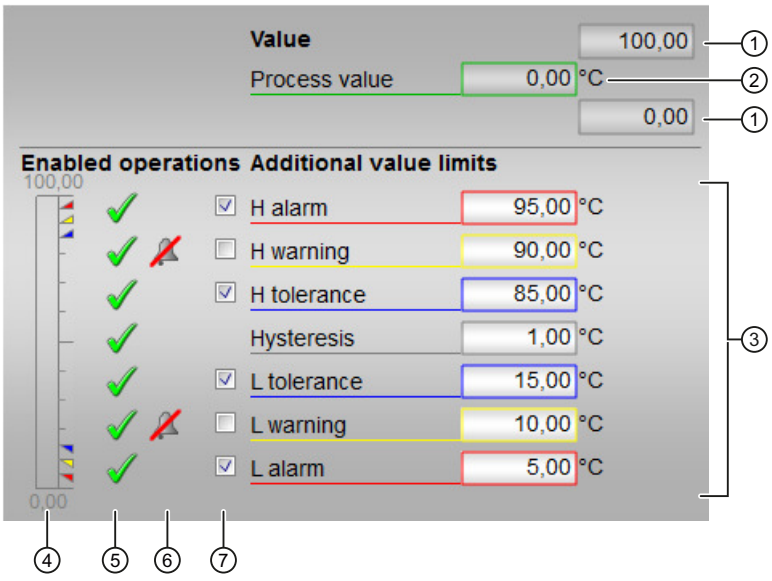
### Limit view of motors

The limit view of motors is only available when an AV block has been interconnected to the motor.

The toolbars of the faceplate and the block icon indicate when the limits are reached or violated.

**Note**

The symbols displayed are not valid for user-configured message classes. Take into consideration the validity of terms for User-configured message classes (Page 45).




### (1) High and low scale range for the additional value

These values provide information on the display range for the bar graph of the additional value. The scale range is defined in the engineering system.

### (2) Display of the additional value including signal status

This area shows the current additional value with the corresponding signal status.

### (3) "Limits for the additional value"

In this area, you can enter the limits for the additional value. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H alarm": Alarm high
- "H warning": Warning high
- "H tolerance": Tolerance high
- Hysteresis
- "L tolerance": Tolerance low
- "L warning": Warning low
- "L alarm": Alarm low

### (4) Bar graph for the additional value

This area shows you the current additional value in the form of a bar graph. You can also see the high low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

You can find additional information about this in the Bar graph (Page 376) section.

### (5) "Enabled operation"

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

The colored triangles indicate the specified limits **(3)** for the additional value.

### (6) Message suppression / delay

Message suppression indicates whether or not the suppression of the associated message in the AS block is activated with the `xx_MsgEn` parameter. The output of messages is not suppressed when the block is installed (all `xx_MsgEn` parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

Alarm delays are also displayed in this position; for more on this see section Area of application of the alarm delays (Page 225).

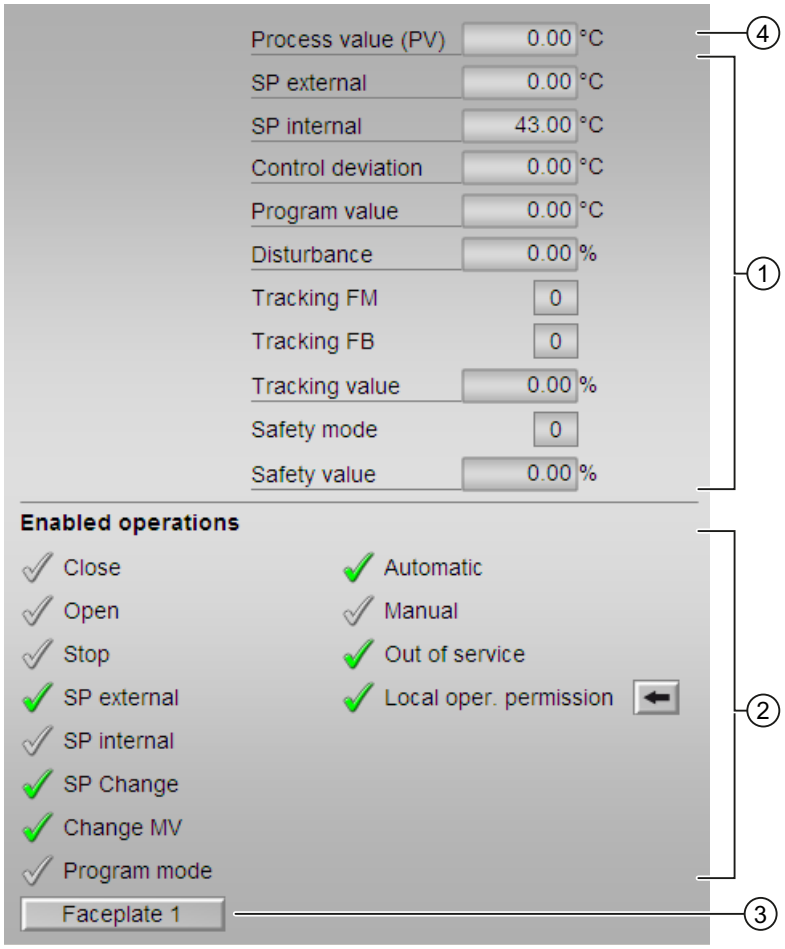
### (7) Suppress messages

You can enable / disable messages by setting the check mark.

## 2.3.18 Preview of FM controllers

### Preview of FM controllers

The preview shows you the parameters that you, as an OS operator, can control in the entire block. You cannot control anything in this view, however.




### (1) Preview area

This area shows you a preview for the following values:

- "SP external": currently applicable external setpoint
- "SP internal": currently applicable internal setpoint
- "Control deviation": Current Control deviation
- "Program mode": Specified value for program mode
- "Disturbance variable": additive value for feedforward control
- "Tracking FM": track a manipulated variable in the FM module (value is 1)
- "Tracking FB": Track manipulated variable at the block (value is 1)

- "Tracking value": Effective manipulated variable for "Track manipulated variable at block"
- "Safety mode": safety mode in the FM module (value is 1)
- "Safety value": effective manipulated variable for "Safety mode"

### (2) "Enabled operations"

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "Close": You can select the manipulated variable "Close". If text is configured for this command, it is also displayed in brackets. You can find more information about this in the Section Labeling of buttons and text (Page 235).
- "Open": You can select the manipulated variable "Open". If text is configured for this command, it is also displayed in brackets. You can find more information about this in the Section Labeling of buttons and text (Page 235).
- "Stop": You can select the manipulated variable "Stop". If text is configured for this command, it is also displayed in brackets. You can find more information about this in the Section Labeling of buttons and text (Page 235).
- "SP external": You can feedforward the external setpoint.
- "SP internal": You can feedforward the internal setpoint.
- "Change SP": You can change the setpoint.
- "Change MV": You can change the manipulated variable.
- "Program mode": You can switch to program mode.
- "Automatic": You can switch to automatic mode.
- "Manual": You can switch to manual mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operating permission": Use the ← button to switch to the standard view of the "OpStations"" block. Additional information is available in the section Operator control permissions (Page 285).

### (3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

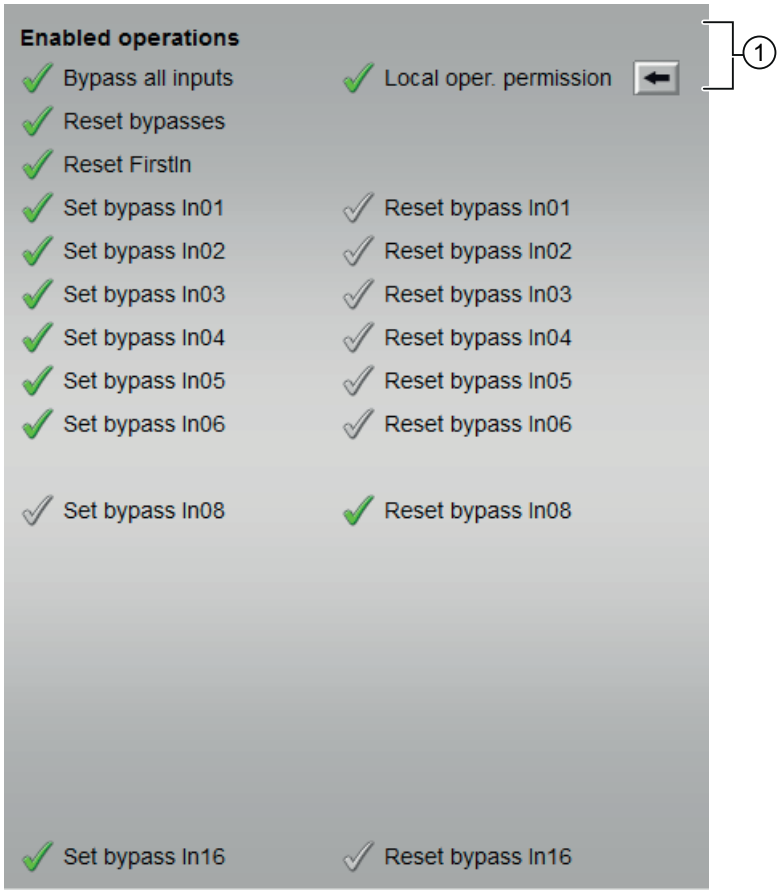
See also the Opening additional faceplates (Page 234) section for more on this.

### (4) Process value

This area displays the real process value (PV).

## 2.3.19 Preview of interlock blocks

### Preview of interlock blocks




### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

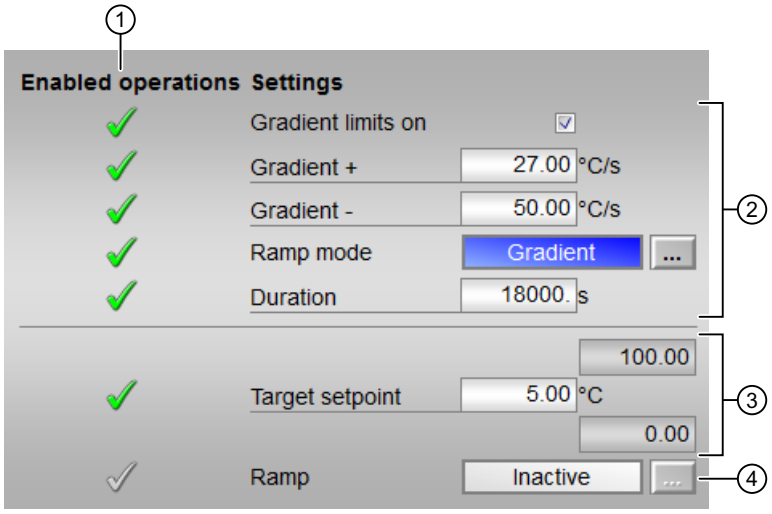
- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

The following enabled operations are shown here:

- "Local operating permission": Use the <-- button to switch to the standard view of the OpStations block. Additional information is available in the section Operator control permissions (Page 285).

## 2.3.20 Ramp view

### Ramp view




### (1) "Enabled operations"

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

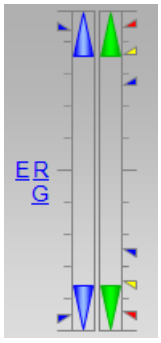
Symbols for enabled operations:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`)

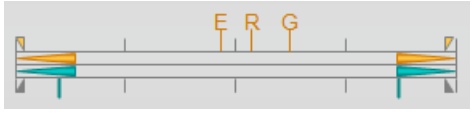
### (2) Enable "gradient limit"

Use this check box to enable "gradient limit" for the setpoint. "Gradient limit" can be set separately for positive or negative setpoint changes ("Gradient +" or "Gradient -"). Refer to the Changing values (Page 291) section for more on this.

If there is a difference between target setpoint and currently effective setpoint, a blue "G" may be displayed at the bar in standard view of FmCont, FmTemp, PIDConL, PIDConR, PIDStepL, OpAnL and MotSpdCL with parameter assignment of `SP_RateTarget` (target setpoint for gradient limit).




If there is a difference between the target manipulated variable and the currently effective manipulated variable, an orange "G" may be displayed at the bar in the standard view of VlvAnl with parameter assignment of `MV_RateTarget` (target manipulated variable for gradient limit).




The gradient limitation includes the Ramp function (Page 131). You can set the ramp mode in the following two ways:

- Gradient
- Duration [s]

### (3) "Target setpoint"

In this area, you can set the type of ramp function for the setpoint.

You can set the time duration and the target setpoint. Refer to the Changing values (Page 291) section for more on this.

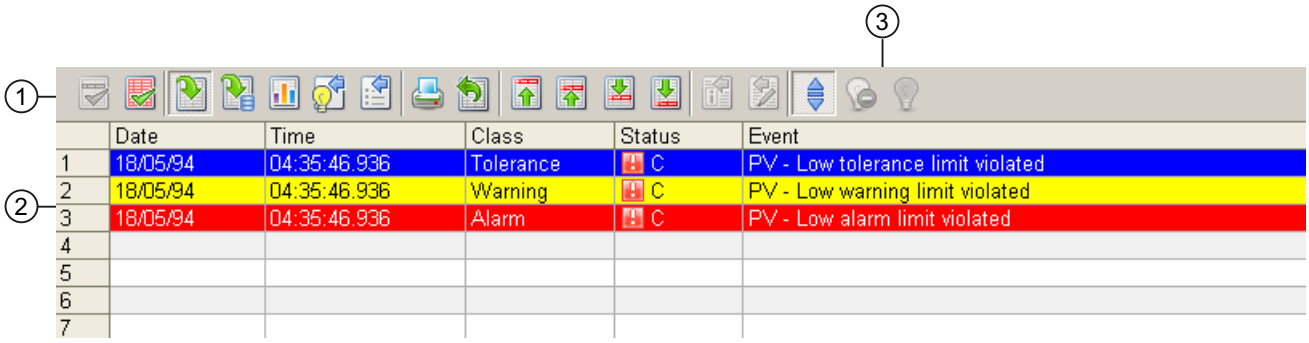
### (4) Enable "Ramp"

You can use this control to enable or disable the configured function in the ramp function for the setpoint change.

You can only enable this when the setpoint specification is set to "Internal" in the standard view of the block. The enable is only valid for one setpoint change and is subsequently disabled again.

## 2.3.21 Alarm view

### Message view




**(1) Toolbar**

If the short-term archive list is selected, a new button appears in the toolbar:



You can use this button to toggle between the "History" and "Operator messages" views.

You must be registered with the "Higher process control" operating permission in order to export and hide messages.

**(2) Display area for alarms**

For additional information about the alarm view, refer to the *WinCC Information System* Online Help.

**(3) "Hide messages" button**

Messages can be displayed and/or hidden with this button. The view of this button changes accordingly:

 Show messages

 Hide messages

"Higher process controlling" operating permission is required and manual hiding must be active. You can find additional information in the manual "Process Control System PCS 7 Operator Station".

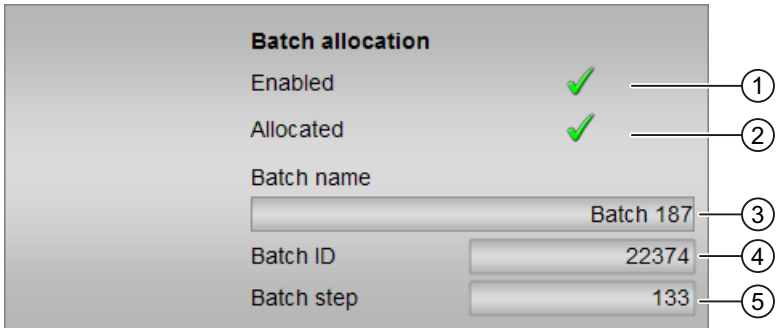
## 2.3.22 Batch view

### Batch view

This area shows a display of the batch that is currently running (`Batchview`).

You can use the internal tag "@APLBatchEnable" to enable/disable the "Batch" button in the toolbar.

| Value of @APLBatchEnable | Operating the "Batch" button in the toolbar | Icon of the "Batch" button in the toolbar |
|---|---|---|
| 0 | Disabled |  |
| 1 | Enabled |  |




### (1) "Enabled"

This area shows you if the block is enabled for operation via SIMATIC BATCH `(BatchEn = 1)`.

### (2) "Allocated"

This area shows if the block is currently in use by SIMATIC BATCH (Occupied = 1).

### (3) "Batch name"

This area shows the name of the batch that is currently running `(Batchname)`.

### (4) "Batch ID"

This area shows the identification number of the batch that is currently running `(BatchID)`.

The batch view is disabled if `BatchID` = 16#00000000.

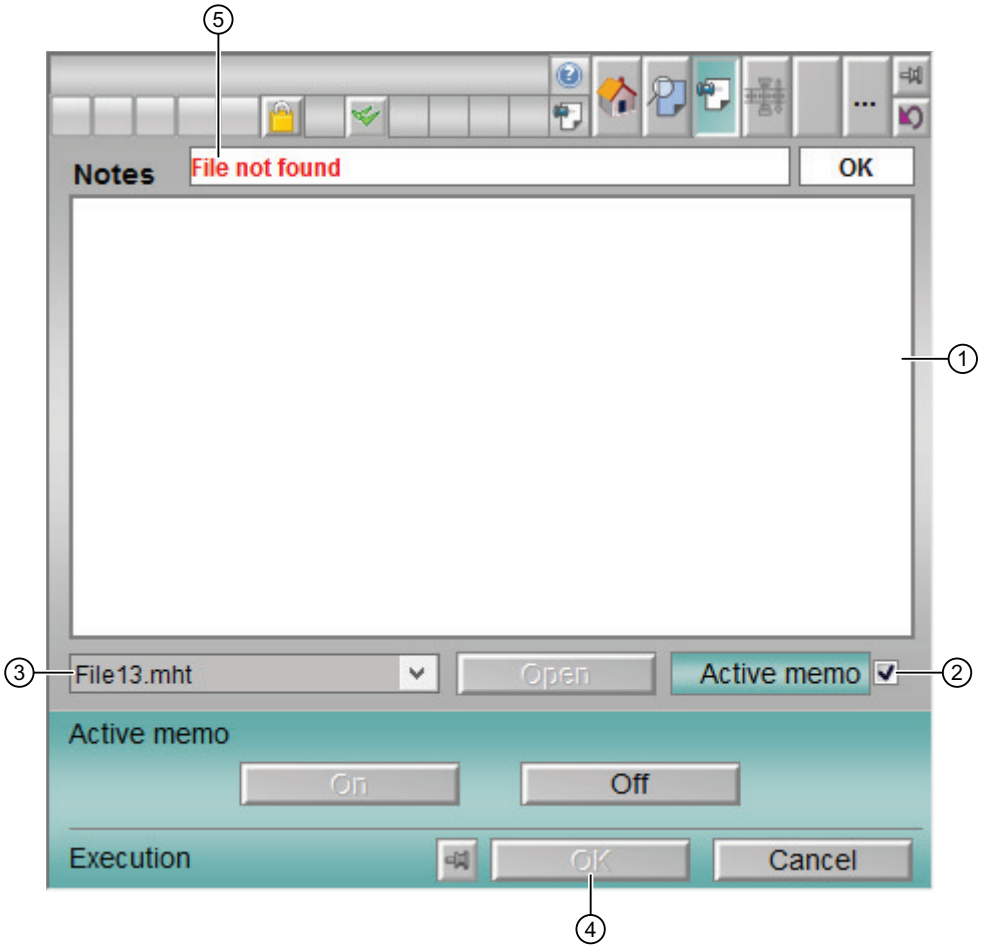
### (5) "Batch step"

This area shows the step number of the batch that is currently running `(StepNo)`.

## 2.3.23 Memo view

### Memo view

You can leave temporary messages for other OS operators in this view. Messages are entered in the text box, and saved and activated by selecting the check box in the lower right corner of the faceplate.




**(1)** Entry field for "Notes"

**(2)** Activating the note

**(3)** Selection of NoteView documents (maximum of three .mht files)

**(4)** Open the selected file and click "OK" to open a new window displaying the selected file

**(5)** Diplays error message

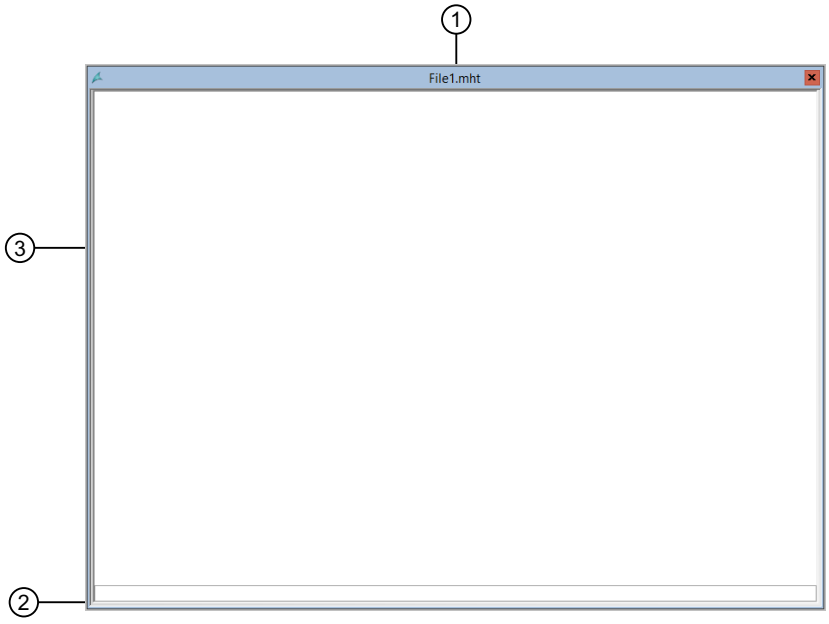
Possible error messages:

- File not found: If the selected file is not avaialbe in the NoteView folder.
- File size exceeded: If the selected file is larger than the allowable limit (> 25 MB).

- Invalid file format: If the file format is not .mht extension.
- File name length exceeded: If the file name exceeds more than 32 characters excluding file extension .mht.

**Note**

- Files that are configured for NoteVew should be saved with UTF-8 encoding. This is applicable to all the intermediate files also, before they are converted to .mht files.
- Jump instruction within the file in NoteView window will not work in the OS Client, Web Server and Web Client.
- No permission levels are required for combo box and open button in the APL_Memo.PDL.
- Document - internal skip marks are not supported.




**(1)** Window Title: indicates the title of Window and name of the file opened as well.

**(2)** Indicates the loading status of the selected MHT document

**(3)** NoteView display area: displays content of the selected file.

Advised maximum size of the file is <= 25 MB for a smooth operation.

**Note**

- This window cannot be resized but it is movable.
- This window is not supported by the "screen composition" functionality.

All the .mht files (NoteView documents) should be placed in the "OS project path\Gracs \NoteView". In case of a Multiproject, standard server(SSM) holds all .mht files and these files are not available in the client. If SSM is not configured then files should be placed in local project folder of client

**Error Handling**

- If selected file is not found in the folder path, an error message is shown as “File not found”.
- If configured file format is other than .mht, an error message is shown as "Invalid file format".
- If selected file is more than 25 MB and if tried to open same, an error is shown as “File size exceeds 25 MB”.
- If file name length exceeds 32 characters, an error is shown as "File name length exceeded 32 characters".

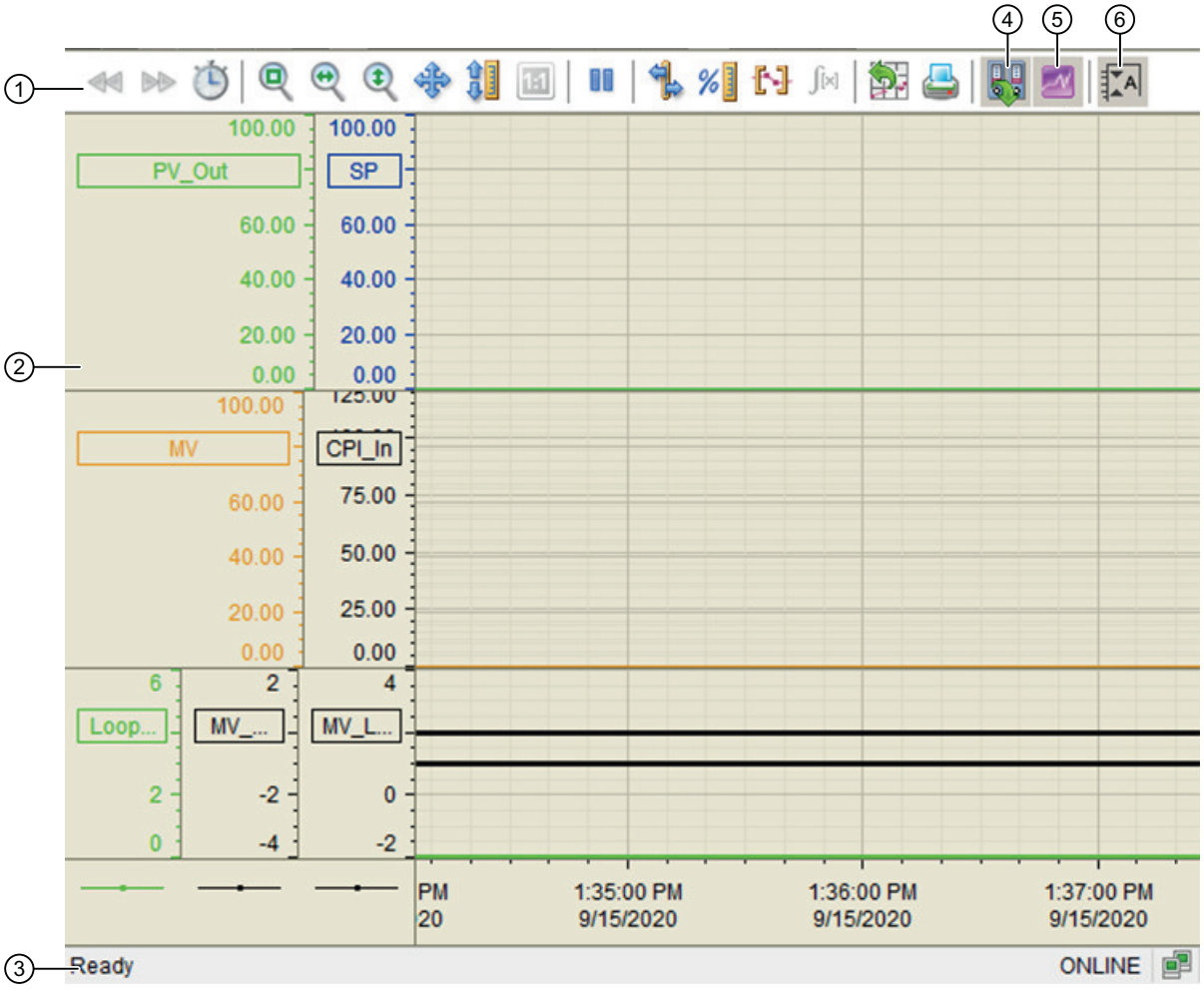
The next time the faceplate is opened or there is a process picture change, you can see in the status bar of the block icon and the faceplate that there is a new message for you.

Clearing the check box deletes the indicators in the status bars.

The message is not deleted automatically.

## 2.3.24 Trend view

### Trend view




**(1)** Toolbar

**(2)** Display area for trends

**(3)** Status bar

**(4)** Button for switching between archive tags and online tags. The status bar shows if the trend view is working with online data or archive data.

**(5)** Button for opening the "Scatter plot" window

(6) Button for "Automatic scaling": This button enables automatic calculation of the range of values. This change is temporary and will revert to original setting when the view is changed, or the faceplate is reopened.

The Export button is only visible and operable with the "Higher-level process control" operating permission.

For additional information about the trend view, refer to the *WinCC Information System* Online Help.

### Configuration of the trend view

The trend view can be configured so that archive values are displayed immediately after opening. A prerequisite for this is that archive variables exist. Proceed as follows:

- A "1" is attached at the block icon in the "TrendPictureName" property. A semicolon is used as the separator to the name of the trend view.
  Example: @pg_apl_trendPID_Statistic.pdl;1

### Special considerations for controllers

You can select two different representations for the display area:

**1. Detailed display (default setting):**

Display area consisting of three coordinate systems:

- Setpoint trend, actual value trend;
- Manipulated variable, control performance index trend;
- Binary trend via automatic/manual, manipulated variable at high or low limit

Open the scatterplot diagram with the user button (number 2) in the toolbar. It shows a coordinate system with the process value on the value axis and the manipulated variable or position feedback on the X axis. A new value pair is entered into the coordinate system with each cycle.

If you want to use the detail display, you need to enter the following in the block icon under Trends in the WinCC Graphics Designer:

TrendPictureName = @pg_apl_trendPID_Statistic.pdl

**2. Simple display:**

Display area consisting of two coordinate systems:

- Setpoint trend, actual value trend;
- Manipulated variable;

If you want to use the simple display, you need to enter the following in the block icon under Trends in the WinCC Graphics Designer:

TrendPictureName =@pg_apl_trendPID.pdl

**Notes on step controllers with position feedback:**

If you use a step controller with position feedback as the controller type, you need to enter the following in the block icon under Trends in the WinCC Graphics Designer:

TrendConfiguration5 = *.MV#Value;...

TrendConfiguration6 = .RbkOut#Value;...

**The following applies to all other controller types (default setting):**

TrendConfiguration5 = .MV#Value;...

TrendConfiguration6 = *.RbkOut#Value;...

## 2.3.25 AddOn View for function blocks

The AddOn view allows you to display an additional view from a connected block in an APL faceplate. This view can either created by yourself or provided by a third-party vendor.

For example, additional values from a field device block connected with the MonAnL block can be displayed in the faceplate.

Messages of the interconnected blocks are also displayed in the "Message" view of the APL faceplate.

**Note**

Alarms for the device specific view can be displayed in message view of the block but there will not be any indication in the overview or block icon.

Configuration:

- To create an additional view, a new picture must be created in the GraCS project directory with the file name @PG_xxx_AddOn.pdl. Where xxx denotes "Blocktype" of the AddOn view. For example, @PG_MyType_AddOn.pdl can be used for displaying an additional view of the "Mytype" block.
- In the CFC, connect the input AddOn of the APL block with any output of the block whose additional view is to be displayed.

After compiling and loading the OS and AS, the additional view in the APL faceplate is available.

Following are the blocks that support additional view:

| AddOn view supported block family | Function blocks |
|---|---|
| Monitoring block for analog values | MonAnL |
| Motor and valve blocks | MotSpdCL |

## 2.3.26 APL Operator Trend Control (AOTC)

### Opening the AOTC window

Press and hold the Ctrl key and left-click the value on the block icon to open the AOTC window. The first value is added to the Trend Control and the detailed information is displayed in the first row of the overview area.

### Adding values to the AOTC window

To add values to the AOTC window, press and hold the Ctrl key and left-click the value on the block icon. The value is added to the Trend Control. You can add up to 8 values in the AOTC window. The following colors are assigned to the trends in the sequence they are added:

1. Black (HTML code 000000)
2. Red (HTML code FF0000)
3. Green (HTML code 00FF00)
4. Blue (HTML code 0000FF)
5. Yellow (HTML code FFFF00)
6. Turkish (HTML code 00FFFF)
7. Pink (HTML code FF00FF)
8. Orange (HTML code FFA100)

If the value is an archived value, the value is displayed from the archive. If the value is not archived, it is displayed as an online value. The default time range for the time axis is 1 minute.

**Note**

For the Web Navigator, when a new value is added, the Trend Control always displays only the online value. You can switch to display the archived values through the buttons "Monday", "Tuesday", and so on.

For information on adding additional values to the AOTC window, refer to **(10) Adding additional values**.

### Adding digital values to the AOTC window

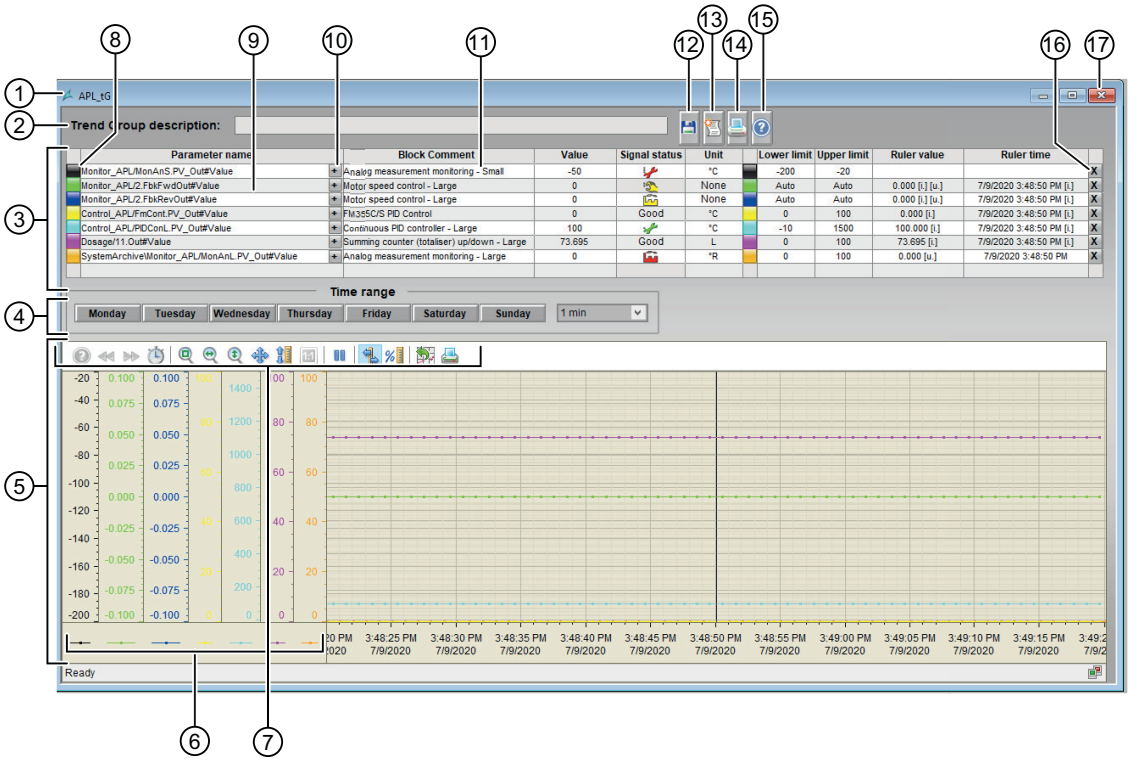
To add digital values to the AOTC window, press and hold the Ctrl key and left-click the value on the block icon. You can add digital values from the following blocks:

| Family | Blocks |
|---|---|
| Drives | MotL, MotRevL, MotSpdCL, MotSpdL, MotS, VlvL, VlvS, Vlv2WayL, VlvAnL, VlvMotL, VlvPosL |
| Operate | OpDi01, OpTrig, OpDi03 |
| Monitor | MonDiL, MonDiS, MonDi08 |

**Example**: If you perform Ctrl key + left-click operation on the MotL block icon, the AOTC window opens showing its current status taken from the parameter `FbkRunOut.Value`. After opening the AOTC window, if you perform Ctrl key + left-click operation on any other block then the parameter corresponding to that block is added to the AOTC window.

For information on adding additional values to the AOTC window, refer to **(10) Adding additional values**.

## The AOTC window




1. Title
2. Trend group description
3. Tag row
4. Time range selection
5. WinCC online trend control
6. Value axis
7. Toolbar of the trend window
8. Enabling/disabling a trend
9. Opening a faceplate
10. Adding additional values
11. Block comment
12. Enabling/disabling a value axis
13. Saving a trend group
14. Opening a separate message window
15. Printing screenshot of the AOTC window
16. Help button

17. Deleting a trend
18. Closing the AOTC window

### (1) Title

The title displays the name of the AOTC. If the trend group is not saved, the default title "APL Operator Trend Control" is displayed. If a saved trend group is opened, title displays the group name.

### (2) Trend group description

This field displays the description of the trend group if it is saved with a description otherwise it is empty.

### (3) Tag row

The tag row displays the following information about the trend added.

- Button to enable/disable the trend:
  Click this button to enable or disable the trend. For more information, refer to the description of **(8) Enabling/disabling a trend**.
- "Parameter name":
  This field displays the name of the trend.
- Button to add additional values:
  Click this button to add additional values to the corresponding parameter. For more information, refer to the description of **(10) Adding additional values**.
- "Value":
  This field displays the current value of the trend.
- "Signal status":
  This field displays the signal status of the trend. The corresponding icon is displayed in case the quality is not good that is the quality code is not 0x80. In case of good quality (quality code 0x80), the text "Good" is displayed.
- "Unit":
  This field displays the unit of the trend. Units are calculated only while adding a trend to the AOTC window. If the unit of a trend changes after it has been added to the AOTC window, the new unit will not be updated in the AOTC window. For example, if `PV_Unit` changes, the trend must be removed and added again.
- Button to enable/disable the value axis:
  Click this button to enable or disable the value axis. For more information, refer to the description of **(11) Enabling/disabling a value axis**.
- "Low limit":
  This field displays the low limit of the trend. The low limit is taken from the tag "XX_OpScale#Low". If the tag is not present, the value is taken automatically by the Trend control and low limit field displays "Auto".

- "High limit":
  This field displays the high limit of the trend. The high limit is taken from the tag "XX_OpScale#High". If the tag is not present, the value is taken automatically by the Trend Control and the high limit field displays "Auto".
- "Ruler value":
  This field displays the ruler value of the trend where the ruler intersects with the trend in the Trend Control.
- "Ruler time":
  This field displays the ruler time of the trend where the ruler intersects with the trend in the Trend Control.
- Button to delete the trend:
  Click this button to delete the trend. For more information, refer to the description of **(15) Deleting a trend**.

### (4) Time range selection

This area provides the option to select the time range based on the weekdays for the archived view and fixed time frame for the online view.

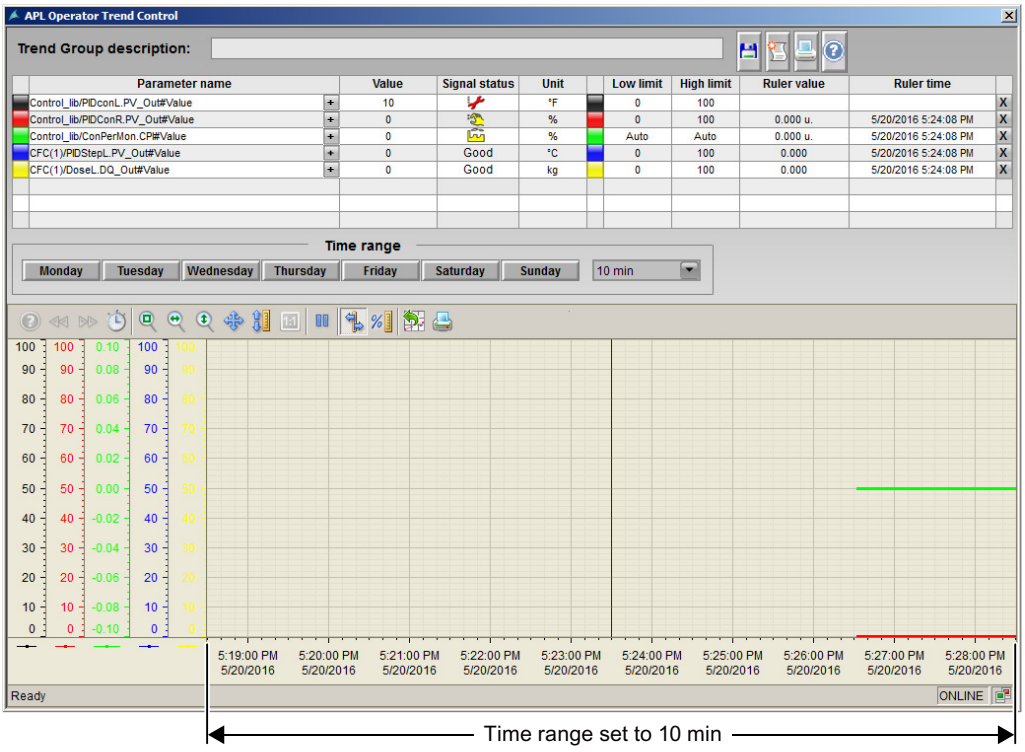
Press the buttons named "Monday" to "Sunday" to switch to the archive view for that day. For example, if today is Tuesday and the "Tuesday" button is pressed, the archived values from today will be displayed. If the "Sunday" button is pressed, the archived values from last Sunday will be displayed. The time range is 24 hours. If the trend is not archived, no value will be displayed.

Press the "Start/stop" button from the trend toolbar to switch back to the online view.

Use the drop-down list in the time range selection area to change the time range of the time axis. The following preconfigured time ranges are available:

- Hour/minute based
  - 1 min
  - 2 min
  - 5 min
  - 10 min
  - 15 min
  - 30 min
  - 1 h
  - 2 h
  - 3 h
  - 5 h
  - 12 h
  - 24 h

The following figure shows the AOTC window when the time range is selected to "10 min" from the drop-down list in "Time range" section:




### (5) WinCC online trend control

The Trend Control displays the trends added to the AOTC window. Each trend added to the AOTC window will appear in different color.

The following colors have been defined default for the 8 trends:

1. Black
2. Red
3. Green
4. Blue
5. Yellow
6. Turkish
7. Pink
8. Orange

### (6) Value axis

Each trend is shown in a separate value axis, therefore there can be maximum of 8 value axis. The value axis appears in the same color as the corresponding trend.

### (7) Toolbar of the trend window

Use these button to operate the trend window.

For more information, refer to *WinCC Information System* Online Help.

## (8) Enabling/disabling a trend

Press these buttons to disabled or enabled trends.

If a trend is enabled, the corresponding button will be displayed in the same color as the value trend.

If a trend is disabled, the corresponding button will be displayed in Grey color.

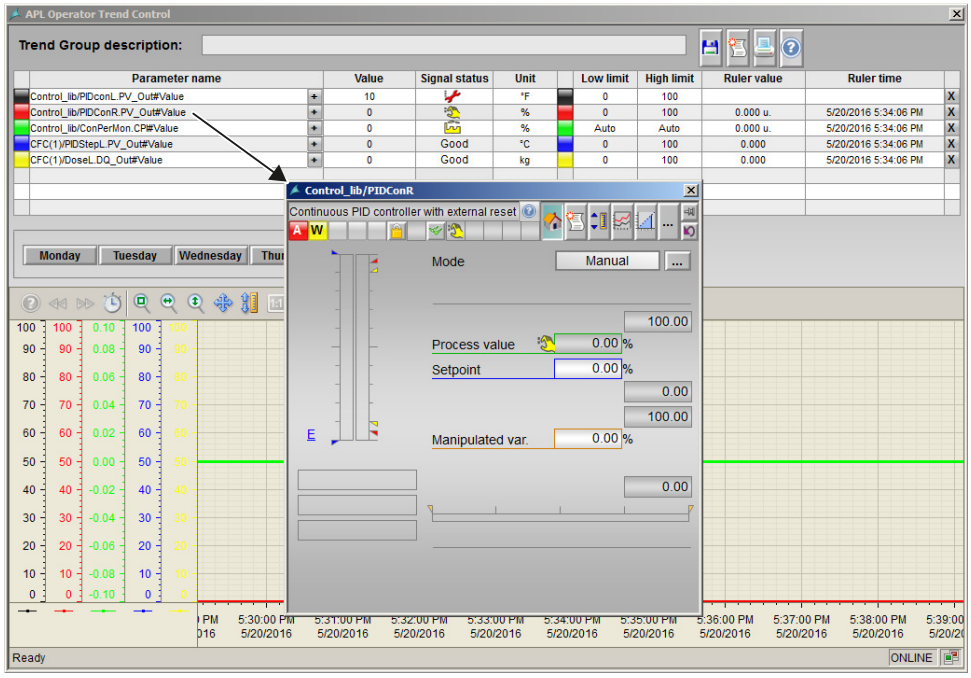
By default the trend added to the AOTC is enabled, therefore the corresponding button is visible and appears in the the same color as the value trend.

When the button is pressed the first time, the corresponding trend is hidden from the Trend Control. The color of the button changes to Grey. The corresponding tag row in the overview area is hidden except the "Parameter name".

On pressing the button again, the corresponding trend is visible again in the Trend Control. The color of the button changes to the same color as the value trend. The corresponding row of values is visible again.

## (9) Opening a related faceplate

Click the parameter name in the tag row to open the corresponding faceplate.

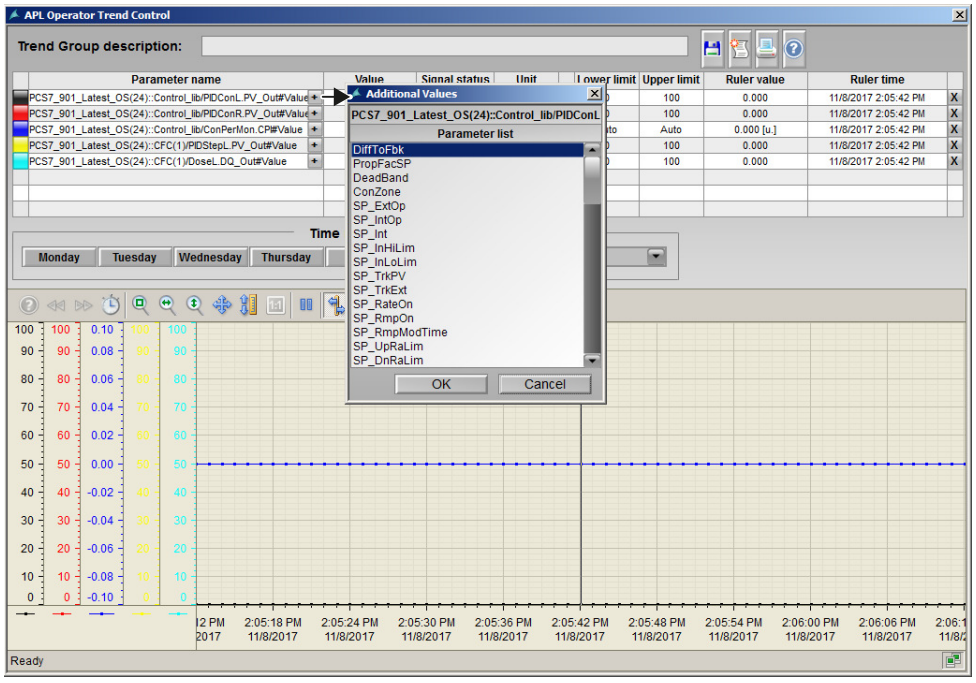



## (10) Adding additional values

This button is used to configure values of a particular function block which is already configured in the parameter list of the AOTC window. This button is visible only for the parameters which are configured in the AOTC window.

Perform the following steps to configure additional values to the AOTC window:

1. Click the [+] button which is available in the right corner of a configured parameter. The "Additional Values" window opens which displays a list of parameters of the corresponding function block.




2. Select a value from the list and click OK to configure the selected value in the AOTC window.

**Note**

- You can select only one value at a time. Multiple selections are disabled.
- Additional value buttons are visible only if the number of trends configured is less than the maximum number of trends allowed (8 values).
- Additional value buttons will be disabled if the configured values in the AOTC reaches its maximum allowed limit.
- Unit of the parameter added from the "Additional Values" window will not be displayed.

## (11) Block Comment

This section provides the name of the block selected corresponding to the selected parameter.

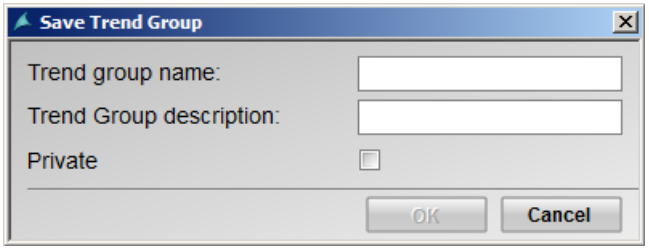
## (12) Enabling/disabling a value axis

Click the button next to the I/O field of the unit to enable or disable the value axis. The value itself will stay in the Trend Control, only the value axis will be hidden from the Trend Control.

## (13) Saving and reopening a trend group

**Saving a trend group:**

1. Click the "Save" button to save the trend group. The "Save Trend Group" window appears to save the trend group:




2. Enter the following details in the "Save Trend Group" window:
   - Trend group name (maximum of 32 characters)
   - Trend group description (maximum of 64 characters)
   - Select the checkbox if you want to save the trend group as private. If the checkbox is not selected (default), the trend group will be saved as public.
     **Saving trend group as private:**
     Saving trend group as private allows only the originator to open the trend again. Private trends cannot be opened by other users. Public trends can be opened and modified by every operator.
3. Press the "OK" button to save the trend group.
   You can abort the operation by pressing the "Cancel" button. The dialog will close without saving the name or description.
   The "OK" button is enabled only if the trend group name is entered and the Enter key is pressed. The trend group descitption is optional.

Once the trend group is saved, further changes made in that trend group from the AOTC window will be saved automatically.

The trend group name should be different from the previously saved trend groups. You cannot save the trend group again with the same name.

The following special characters are not allowed in the trend group name:

' ' (space character), '.', ',', ';', ':', '/', '=', '\t', '\\', '"', '<', '>', '{', '}', '[', ']', '|', '?', '*'

**Opening a trend group:**

Click the "Trend system" standard button at the bottom of the WinCC runtime window to open a saved trend.



**Opening trend groups in "Full screen display":**

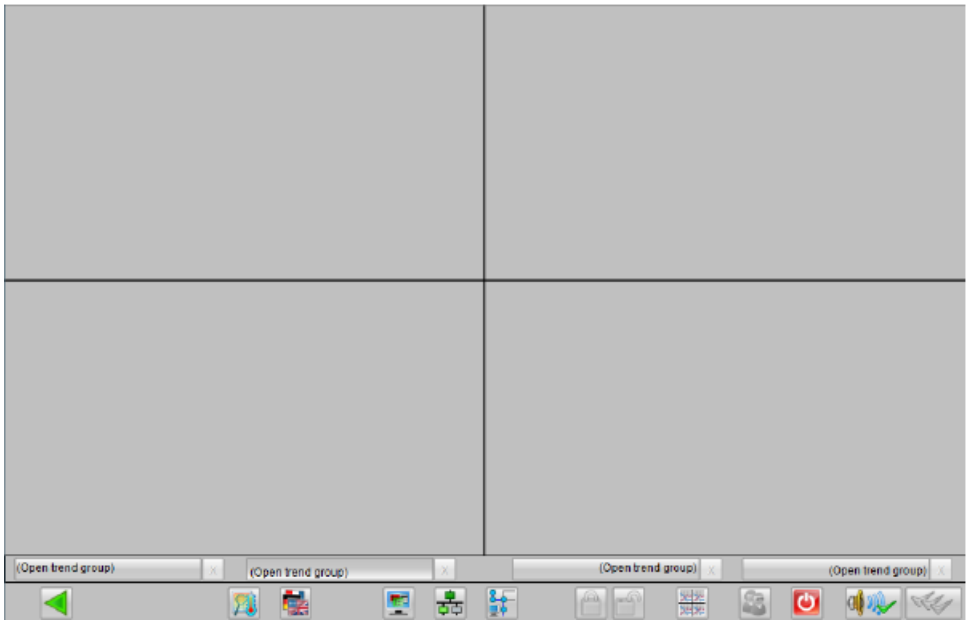
The AOTC full screen display is used to display configured trend groups on a dedicated screen. The AOTC full screen window has four quadrants which means you can display four different trend groups at a time. This window is termed as “Quad trend window” or “Full screen display”.

Perform the following steps to display the trend groups in "Full screen display":

1. Click the AOTC full screen display button at the bottom of the WinCC runtime window.



The AOTC full screen display opens with four empty quadrants and with four buttons at the bottom to select a saved trend group for each quadrant.



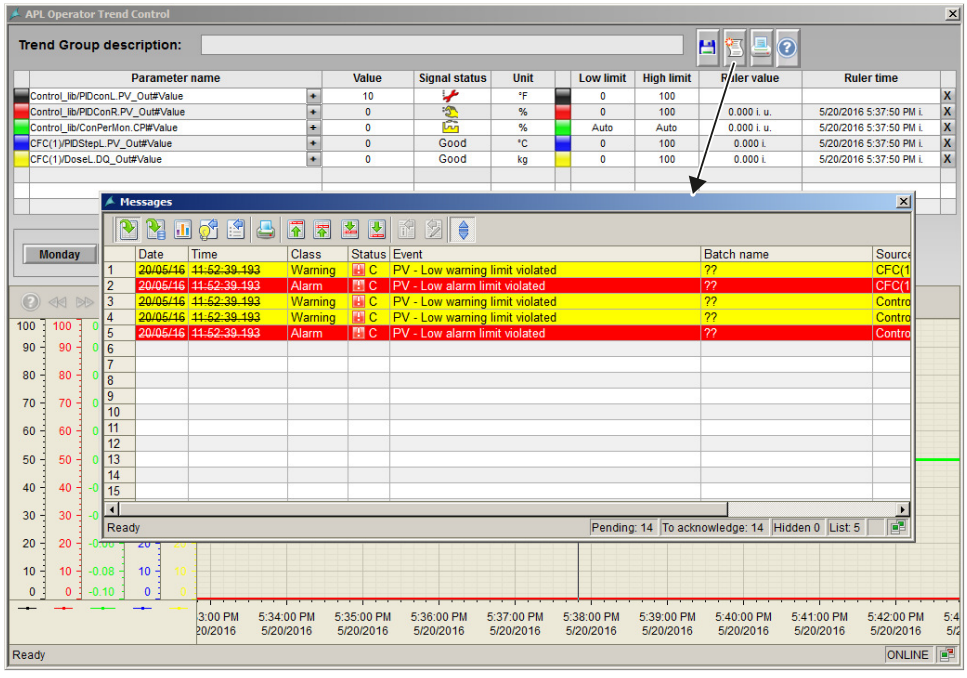

2. Click the "(open trend group)" button at the bottom of the AOTC full screen display for the quadrant in which you want to display a particular trend group. A window opens with a list of saved trend groups.

3. Select a trend group from the list and click "Display" to load the trend group into the corresponding quadrant.
4. Use all four buttons to load four different trend groups into the different quadrants of the full screen display.




## (14) Opening a separate message window

Press the "Messages" button to open a new message window which displays the messages for the selected block types corresponding to the trends added in the AOTC window.



In the online mode, only the actual messages (all active messages, acknowledged and unacknowledged) will be displayed.

If you select a time range displaying values from the archive, the message window will display all the messages within this time range.

Messages only from the enabled trend will be displayed. If a trend is enabled or disabled and the message window is open, the corresponding messages will be displayed or not displayed accordingly. Similarly if a trend is deleted, the corresponding messages will disappear from the messages window.

If two or more trends of the same block type are added to the AOTC window, and one of them is disabled, the corresponding messages will not be removed from message window. All the trends should be disabled or deleted to remove the corresponding messages from the message window.

Only one message window can be opened at a time. If one more AOTC window is opened and you want to display the messages for the values in that window, the already opened message view should be closed and opened again from the specific AOTC window.

### (15) Printing screenshot of the AOTC window

Press the "Print" button to print the screenshot of the complete AOTC window. WinCC runtime (also the Web Navigator) should be in full screen mode without scroll bar. As screenshot is taken from the process picture visible in runtime, the AOTC window should be visible completely in the runtime, otherwise the screenshot will show only the visible section of the AOTC faceplate.

### (16) Help button

Click this button to open the AOTC chapter in the *APL Operator Guide* online help. You will find detailed information about the AOTC window and its functionalities in this online help.

### (17) Deleting a trend

Click the "X" button at the end of the row in the overview area to delete existing values.

The corresponding trend is removed from the Trend Control. In the overview area, the rows below the deleted row are moved up and an empty row is added at the end.

New values are added to the first empty row.

### (18) Closing the AOTC window

Click the "Close" button in the title bar to close the AOTC window.

### Operator permissions

The operator permissions are similar to the existing online trend function.

The following fields are enabled only if the area authorization is available:

1. Delete buttons to delete trends.
2. Save button to open the pop-up window to save the trend group.

3. Field to enter the trend group name in the pop-up window.
4. Field to enter the trend group description in the pop-up window.
5. Checkbox to select private trend group.
6. The "OK" button to save the trend in the pop-up window.

**Note**

The functions for example, disable/enable trend, disable/enable value axis, delete trend, open message window, print AOTC screenshot, select private trend group, need no operation area or quitting function. The feature "Save Trend Group" needs an operation area where only 3 step operation is possible. It does not support 2 step operation.

## 2.3.27 Limit operation and display in the faceplate

### Limit operation and display in the faceplate

The limit value view of the faceplate can be used to modify limits and the hysteresis if the corresponding operating permission (higher process controlling) is available. The limits are displayed graphically in the standard view of the faceplate.

If the limits are reached or exceeded, an alarm, warning or tolerance class message is triggered. This is indicated graphically as follows:

| Symbol | Meaning |
|---|---|
|  | Alarm |
|  | Warning |
|  | Tolerance |

**Note**

The text displayed at the labels for the limit fields in limit view depends on the setting of the PCS 7 message system.

If the “User-configured message classes (Page 45)” is activated then the displayed text is taken from the “TextOp” property of the limit object else it is taken from property “AnaText”

## 2.3.28 Central color management

### Central color management

As of version 8.1, the colors in the faceplates and block icons can be changed centrally via the central color palette in WinCC. The central color palette is located in the "User interface and design" tab of the project properties.

The following colors cannot be changed via the central color palette:

- The colors (except trend colors) of WinCC Alarm Control, WinCC Online Trend Control, WinCC Ruler Control and WinCC Function Control.
- The window titles can only be changed by editing the Windows designs.
- The colors of check boxes cannot be changed.
- Alarm colors are changed via the PCS 7 message system.
- Trend colors are not automatically changed in the block icons in existing projects, because these properties are changeable instance-specific and not in the update mechanisms for the block icons.

The user can use the color range of 0-199; the APL color range begins with the color index 200.

Description of the APL color index:

### Color index for block icons

The following color indexes can be created for block icons.

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Type block icons |
|---|---|---|---|---|---|---|---|
| 200 | 1 | 0 | B6B6B6 | Gray | Background: Background color | BackgroundColor | Block icons with analog values |
| 201 | 1 | 0 | DADADA | Very light gray | Background: Fill pattern color | BackgroundFillPatternColor | Block icons with analog values |
| 202 | 1 | 0 | B6B6B6 | Gray | Tag display: Background color | TagBackColor | All block icons with tag display |
| 203 | 1 | 0 | DADADA | Very light gray | Tag display: Fill pattern color | TagFillColor | All block icons with tag display |
| 204 | 1 | 0 | 919191 | Dark gray | Tag display: Border color | TagBorderColor | All block icons with tag display |
| 205 | 1 | 0 | 000000 | Black | Tag display: Font color | TagFontColor | All block icons with tag display |
| 206 | 1 | 0 | 00FFFA | Pure cyan | Tag highlight: Background color | BlockIconHighlightBackColor | All block icons |
| 207 | 1 | 0 | 5CC2BC | Medium cyan | Tag highlight: Fill pattern color | BlockIconHighlightFillColor | All block icons |
| 208 | 1 | 0 | FFFFFF | White | Tag highlight: Font color | BlockIconHighlightFontColor | All block icons |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Type block icons |
|---|---|---|---|---|---|---|---|
| 209 | 1 | 0 | B6B6B6 | Gray | Display of high limit: Background color | LimitObjectHighBackColor | Block icons with display of the high limit |
| 210 | 1 | 0 | DADADA | Very light gray | Display of high limit: Fill pattern color | LimitObjectHighFillColor | Block icons with display of the high limit |
| 211 | 1 | 0 | B6B6B6 | Gray | Display of the low limit: Background color | LimitObjectLowBackColor | Block icons with display of the low limit |
| 212 | 1 | 0 | DADADA | Very light gray | Display of the low limit: Fill pattern color | LimitObjectLowFillColor | Block icons with display of the low limit |
| 213 | 1 | 0 | B6B6B6 | Gray | Empty rectangle: Background color | EmptyRectBackColor"[1] | Block icons with status bar |
| 214 | 1 | 0 | DADADA | Very light gray | Empty rectangle: Fill pattern color | EmptyRectFillColor[1] | Block icons with status bar |
| 215 | 1 | 0 | 808080 | Dark gray | Empty rectangle: Border color | EmptyRectBorderColor[1] | Block icons with status bar |
| 216 | 1 | 0 | FFFFFF | White | Process value: Background color | Value1BackColorValue10BackColor | Block icons with analog value |
| 217 | 1 | 0 | 000000 | Black | Process value: Font color | Value1FontColorValue10FontColor | Block icons with analog value |
| 218 | 1 | 100 | C0C0C0 | None (transparent) | Setpoint: Background color | Value2BackColor | Block icons with analog value |
| 219 | 1 | 0 | 0000FF | Pure blue | Setpoint: Font color | Value2FontColor | Block icons with analog value |
| 220 | 2 | 100 | C0C0C0 | None (transparent) | Changed variable: Background color | Value3BackColor | Block icons with analog value |
| 221 | 2 | 0 | F24F00 | Pure orange | Changed variable: Font color | Value3FontColor | Block icons with analog value |
| 222 | 2 | 100 | C0C0C0 | None (transparent) | Read-back value: Background color | Value3BackColor Value4BackColor | Block icons with analog value |
| 223 | 2 | 0 | 008080 | Dark cyan | Read-back value: Font color | Value3FontColor Value4FontColor | Block icons with analog value |
| 224 | 2 | 100 | B6B6B6 | None (transparent) | Unit[2], process value/ setpoint: Background color | UnitBackColor, Unit1BackColor-Unit4BackColor | Block icons with analog value |
| 225 | 2 | 0 | 494949 | Very dark gray | Unit, process value/ setpoint: Font color | UnitFontColor, Unit1FontColor-Unit4FontColor | Block icons with analog value |
| 226 | 2 | 100 | B6B6B6 | None (transparent) | Unit[2], changed variable: Background color | Unit2BackColor | Block icons with analog value |
| 227 | 2 | 0 | 494949 | Very dark gray | Unit, changed variable: Font color | Unit2FontColor | Block icons with analog value |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Type block icons |
|---|---|---|---|---|---|---|---|
| 228 | 2 | 100 | B6B6B6 | None (transparent) | Unit[2], read-back value: Background color | Unit1BackColor, Unit2BackColor | Block icons with analog value |
| 229 | 2 | 0 | 494949 | Very dark gray | Unit, read-back value: Font color | Unit1FontColor, Unit2FontColor | Block icons with analog value |
| 230 | 2 | 0 | 6D6D6D | Very dark gray | "Not used" state: background color | BackColorOn1 | Interlock |
| 231 | 2 | 0 | 919191 | Dark gray | "Not used" state: fill pattern color | FillColorOn1 | Interlock |
| 232 | 2 | 0 | 0000FF | Pure blue | "Bypassed" state: background color | BackColorOn2 | Interlock |
| 233 | 2 | 0 | 00A2E8 | Pure blue | "Bypassed" state: fill pattern color | FillColorOn2 | Interlock |
| 234 | 2 | 0 | E4D400 | Pure yellow | "Simulation" state: background color | BackColorOn3 | Interlock |
| 235 | 2 | 0 | FFFF00 | Pure yellow | "Simulation" state: fill pattern color | FillColorOn3 | Interlock |
| 236 | 2 | 0 | 00B500 | Bright lime green | "Unlocked" state: background color | BackColorOn4 | Interlock |
| 237 | 2 | 0 | 00FF00 | Pure lime green | "Unlocked" state: fill pattern color | FillColorOn4 | Interlock |
| 238 | 2 | 0 | E60000 | Pure red | "Locked" state: background color | BackColorOn5 | Interlock |
| 239 | 2 | 0 | FF0000 | Pure red | "Locked" state: fill pattern color | FillColorOn5 | Interlock |
| 240 | 3 | 0 | 00B500 | Bright lime green | "On" state: Background color | BackColorOn, OnBackColor | Family counter / MonDi08/ MonDiL-MonDiS (type 1 and 4) / OpDi03 / ShrdResS/ TimeTrig/ TotalL |
| 241 | 3 | 0 | 00FF00 | Pure lime green | "On" state: Fill pattern color | FillColorOn, OnFillColor | |
| 242 | 3 | 0 | 00FF00 | Pure lime green | "On" state: Background color | Value2BackColor | OpDi01, OpTrig |
| 243 | 3 | 0 | 000000 | Black | "On" state: Font color | Value2FontColor | OpDi01, OpTrig |
| 244 | 3 | 0 | FFFFFF | White | "On" state: Background color | Value2BackColor | MonDiL-MonDiS (types 2, 3 and 5) |
| 245 | 3 | 0 | 000000 | Black | "On" state: Font color | Value2FontColor | MonDiL-MonDiS (types 2, 3 and 5) |
| 246 | 3 | 0 | FFFFFF | White | "On" state: Font color | FontColorOn | MonDiL-MonDiS (type 4) |
| 247 | 3 | 0 | B6B6B6 | Gray | "Off" state: Background color | BackColorOff, BackColorOn | |
| 248 | 3 | 0 | DADADA | Very light gray | "Off" state: Fill pattern color | FillColorOff, FillColorOn | |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Type block icons |
|---|---|---|---|---|---|---|---|
| 249 | 3 | 0 | DADADA | Very light gray | "Off" state: Background color | BackColorOff | Family counter, MonDi08, TimeTrig |
| 250 | 3 | 0 | FFFFFF | White | "Off" state: Fill pattern color | FillColorOff | Family counter, MonDi08, TimeTrig |
| 251 | 3 | 0 | FFFFFF | White | "Off" state: Background color | Value1BackColor | MonDiLMonDiS (types 2, 3 and 5) |
| 252 | 3 | 0 | 000000 | Black | "Off" state: Font color | Value1FontColor | MonDiLMonDiS (types 2, 3 and 5) |
| 253 | 3 | 0 | 000000 | Black | "Off" state: Font color | FontColorOff | MonDiLMonDiS (type 4) |
| 254 | 3 | 0 | B6B6B6 | Gray | "Not used" state: Background color | BackColorNA | MonDi08 |
| 255 | 3 | 0 | 000000 | Black | "No operation" state: Font color | ForeColorValue | OpStations |
| 256 | 3 | 0 | C0C0C0 | Light gray | Display of the selected analog value: Background color | Value2BackColor | SelA16In |
| 257 | 3 | 0 | 6D6D6D | Very dark gray | Display of the selected analog value: Font color | Value2FontColor | SelA16In |
| 258 | 3 | 0 | 494949 | Very dark gray | ISO/DIN symbols: Line color | BorderColor | |

[1] New property

[2] Note: The pattern in the block icon must be changed to "single color" for the color to take effect.

## Color indexes for faceplates

The following color indexes can be created for faceplates.

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Types views/ objects |
|---|---|---|---|---|---|---|---|
| 259 | 3 | 0 | AFAFAF | Gray | View: Background color | BackColor | Views in all faceplates |
| 260 | 4 | 0 | DEDEDE | Very light gray | View: Fill pattern color | FillColor | Views in all faceplates |
| 261 | 4 | 0 | 62ACAA | Dark cyan | Operating area: Background color | BackColor | Total operating area |
| 262 | 4 | 0 | AACFCD | Gray cyan | Operating area: Fill pattern color | FillColor | Total operating area |
| 263 | 4 | 0 | 000000 | Black | Title or label of operating area: Font color | ForeColor | All static texts in the operating area |
| 264 | 4 | 0 | C0C0C0 | Light gray | Slider operating area: Background color | BackColor, BackColorTop, BackColorBottom, ButtonColor, HighLimitColor, LowLimitColor | Operating area "@PG_APL_OA_Analogwithlimits100.PDL" |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Types views/ objects |
|---|---|---|---|---|---|---|---|
| 265 | 4 | 0 | FFFFFF | White | Slider operating area: Border background color | BorderBackColor | Operating area "@PG_APL_OA_Analogwithlimits100.PDL" |
| 266 | 4 | 0 | 000000 | Black | Slider operating area: Border color | BorderColor | Operating area "@PG_APL_OA_Analogwithlimits100.PDL" |
| 267 | 4 | 0 | DADADA | Very light gray | Block comment: Fill pattern color | FillColor | Header in all faceplates |
| 268 | 4 | 0 | 808080 | Dark gray | Block comment: Border color | BorderColor | Header in all faceplates |
| 269 | 4 | 0 | 000000 | Black | Block comment: Font color | FontColor | Header in all faceplates |
| 270 | 4 | 0 | B6B6B6 | Gray | Empty rectangle overview: Background color | BackColor | Header in all faceplates |
| 271 | 4 | 0 | DADADA | Very light gray | Empty rectangle overview: Fill pattern color | FillColor | Header in all faceplates |
| 272 | 4 | 0 | 808080 | Dark gray | Empty rectangle overview: Border color | BorderColor | Header in all faceplates |
| 273 | 4 | 0 | C0C0C0 | Light gray | Group display overview: Background color | BackColor | Header in all faceplates [1] |
| 274 | 4 | 0 | C0C0C0 | Light gray | Toolbar button: Background color | BackColor | Header in all faceplates |
| 275 | 4 | 0 | BABABA | Gray | Toolbar button: Fill pattern color | FillColor | Header in all faceplates |
| 276 | 4 | 0 | 808080 | Dark gray | Toolbar button: Border color | BorderColor | Header in all faceplates |
| 277 | 4 | 0 | 000000 | Black | Toolbar button: Font color | FontColor | Header in all faceplates [2] |
| 278 | 4 | 0 | C0C0C0 | Light gray | Header button[3]: Background color | BackColor | Header in all faceplates |
| 279 | 4 | 0 | 808080 | Dark gray | Toolbar button: Border color | BorderColor 8 | Header in all faceplates |
| 280 | 5 | 0 | 000000 | Black | Title or label in views: Font color | TextFontColor, FontColor, ForeColor, LeftHandTextFontColor[4] | Views with static text and APL objects with type ID=17-20, 22-24, 27, 29, 64, 120, 130 |
| 281 | 5 | 0 | AFAFAF | Gray | Button in views/operating areas: Background color | BackColor | Views with APL objects with type ID=30-34, 50-52 |
| 282 | 5 | 0 | CECECE | Light gray | Button in views/operating areas: Fill pattern color | FillColor | Views with APL objects with type ID=30-34, 50-52 |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Types views/ objects |
|---|---|---|---|---|---|---|---|
| 283 | 5 | 0 | 808080 | Dark gray | Button in views/operating areas: Border color | BorderColor | Views with APL objects with type ID=30-34, 50-52 |
| 284 | 5 | 0 | FFFFFF | White | Button in views/operating areas: 3D border color | BorderColorTop | Views with APL objects with type ID=30-34, 50-52 |
| 285 | 5 | 0 | 808080 | Dark gray | Button in views/operating areas: 3D shadow color | BorderColorBottom | Views with APL objects with type ID=30-34, 50-52 |
| 286 | 5 | 0 | 000000 | Black | Button in views/operating areas: Font color | ForeColor | Views with APL objects with type ID=30-34, 50-52 |
| 287 | 5 | 0 | DADADA | Very light gray | Analog value enabled in views/operating areas: Background color | EnabledBackColor, BackColor, BackColor_OK | Views with I/O fields (also extended) and APL objects with type ID=20, 22-24, 27, 28-29, 130 |
| 288 | 5 | 0 | FFFFFF | White | Analog value enabled in views/operating areas: Fill pattern color | EnabledFillColor, FillColor, FillColor_OK | Views with I/O fields (also extended) and APL objects with type ID=20, 22-24, 27, 28-29, 130 |
| 289 | 5 | 0 | B6B6B6 | Gray | Analog value disabled in views/operating areas: Background color | DisabledBackColor, BackColor, BackColor_OK | Views with I/O fields (also extended) and APL objects with type ID=20, 22-24, 27, 28-29, 91[13],92,130 |
| 290 | 5 | 0 | DADADA | Very light gray | Analog value disabled in views/operating areas: Fill pattern color | DisabledFillColor, FillColor, FillColor_OK | Views with I/O fields (also extended) and APL objects with type ID=20, 22-24, 27, 28-29, 91[13],92,130 |
| 291 | 5 | 0 | 000000 | Black | Analog value or unit in views/operating areas: Font color | ValueFontColor, ForeColor, ForeColor_OK | Views with I/O fields (also extended) and APL objects with type ID=16, 20, 22-24, 27, 28-29, 91, 130 |
| 292 | 5 | 0 | 00B500 | Bright lime green | Analog process value: Border color | LineColor, BorderColor | Views with I/O fields (also extended) and APL objects with type ID=20, 24 |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Types views/ objects |
|---|---|---|---|---|---|---|---|
| 293 | 5 | 0 | 0000FF | Pure blue | Analog setpoint: Border color | LineColor, BorderColor | Views with I/O fields (also extended) and APL objects with type ID=20, 28 |
| 294 | 5 | 0 | D27A00 | Bright orange | Changed analog variable: Border color | LineColor, BorderColor | Views with I/O fields (also extended) and APL objects with type ID=20, 24, 28 |
| 295 | 5 | 0 | 008582 | Dark cyan | Analog read-back value: Border color | LineColor, BorderColor | Views with I/O fields (also extended) and APL objects with type ID=20 |
| 296 | 5 | 0 | 808080 | Dark gray | Analog parameter/ range value: border color | LineColor, BorderColor | Views with I/O fields (also extended) and APL objects with type ID=20, 22-24, 27, 28, 90, 92, 130 |
| 297 | 5 | 0 | 000000 | Black | Analog limit (no alarm), input value, offset, gradient: border color | LineColor, BorderColor | In "@PG_CntOhSc_Limit.PDL", "@PG_MonAnL_Standard.PDL", "@PG_Ratio_Standard.PDL", "@PG_TotalL_Standard.PDL" APL objects with type ID=20, 24, 28, 29 and HMIStaticText |
| 298 | 5 | 0 | 808080 | Dark gray | Empty analog value displays | BorderColor | Views with APL objects with type ID=21, 25, 26 |
| 299 | 5 | 0 | 00B500 | Bright lime green | Binary value on: Background color | OnBackColor, State1BackColor | Views with APL objects with type ID=17,18[13],120 |
| 300 | 6 | 0 | 00FF00 | Pure lime green | Binary value on: Fill pattern color | OnFillColor, State1FillColor | Views with APL objects with type ID=17,18[13],120 |
| 301 | 6 | 0 | FFFFFF | White | Binary value on: Font color | OnFontColor, State1FontColor | Views with APL objects with type ID=17,18[13],120 |
| 302 | 6 | 0 | FFFFFF | White | Binary value off: Background color | OffBackColor, State2BackColor | Views with APL objects with type ID=17,18[13],120 |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Types views/ objects |
|---|---|---|---|---|---|---|---|
| 303 | 6 | 0 | DADADA | Very light gray | Binary value off: Fill pattern color | OffFillColor, State2FillColor | Views with APL objects with type ID=17,18[13],120 |
| 304 | 6 | 0 | 000000 | Black | Binary value off: Font color | OffFontColor, State2FontColor | Views with APL objects with type ID=17,18[13],120 |
| 305 | 6 | 0 | 808080 | Dark gray | Binary value: Border color | BorderColor | Views with APL objects with type ID=17,18[13],120 |
| 306 | 6 | 0 | B6B6B6 | Gray | Binary value preview: Background color | BackColor | Preview, APL objects with type ID=14 |
| 307 | 6 | 0 | DADADA | Very light gray | Binary value preview: Fill pattern color | FillColor | Preview, APL objects with type ID=14 |
| 308 | 6 | 0 | 000000 | Black | Binary value preview: Font color | FontColor | Preview, APL objects with type ID=14 |
| 309 | 6 | 0 | 808080 | Dark gray | Binary value preview: Border color | BorderColor | Preview, APL objects with type ID=14 |
| 310 | 6 | 0 | 00B500 | Bright lime green | Status display "On" mode: background color | State1BackColor... State7BackColor | Standard view, APL objects with type ID=11 |
| 311 | 6 | 0 | 00FF00 | Pure lime green | Status display "On" mode: fill pattern color | State1FillColor... State7FillColor | Standard view, APL objects with type ID=11 |
| 312 | 6 | 0 | FFFFFF | White | Status display "On" mode: font color | State1FontColor... State7FontColor | Standard view, APL objects with type ID=11 |
| 313 | 6 | 0 | FFFFFF | White | Status display "Manual" mode: background color | State1BackColor... State7BackColor | Standard view, APL objects with type ID=11 |
| 314 | 6 | 0 | CECECE | Light gray | Status display "Manual" mode: fill pattern color | State1FillColor... State7FillColor | Standard view, APL objects with type ID=11 |
| 315 | 6 | 0 | 000000 | Black | Status display "Manual" mode: font color | State1FontColor... State7FontColor | Standard view, APL objects with type ID=11 |
| 316 | 6 | 0 | 00FF00 | Pure lime green | Status display "Automatic" mode: background color | State1BackColor... State7BackColor | Standard view, APL objects with type ID=11 |
| 317 | 6 | 0 | 008500 | Dark lime green | Status display "Automatic" mode: fill pattern color | State1FillColor... State7FillColor | Standard view, APL objects with type ID=11 |
| 318 | 6 | 0 | FFFFFF | White | Status display "Automatic" mode: font color | State1FontColor... State7FontColor | Standard view, APL objects with type ID=11 |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Types views/ objects |
|---|---|---|---|---|---|---|---|
| 319 | 6 | 0 | 00CECA | Bright cyan | Status display "Local" mode: background color | State1BackColor... State7BackColor | Standard view, APL objects with type ID=11 |
| 320 | 7 | 0 | 00FFFA | Pure cyan | Status display "Local" mode: fill pattern color | State1FillColor... State7FillColor | Standard view, APL objects with type ID=11 |
| 321 | 7 | 0 | 000000 | Black | Status display "Local" mode: font color | State1FontColor... State7FontColor | Standard view, APL objects with type ID=11 |
| 322 | 7 | 0 | 0000FF | Pure blue | Status display "Programm MV/SP" mode: background color | State1BackColor... State7BackColor | Standard view, APL objects with type ID=11 |
| 323 | 7 | 0 | 98BDFF | Very light blue | Status display "Programm MV/SP" mode: fill pattern color | State1FillColor... State7FillColor | Standard view, APL objects with type ID=11 |
| 324 | 7 | 0 | FFFFFF | White | Status display "Programm MV/SP" mode: font color | State1FontColor... State7FontColor | Standard view, APL objects with type ID=11 |
| 325 | 7 | 0 | 919191 | Dark gray | Status display "Out of service" mode: background color | State1BackColor... State7BackColor | Standard view, APL objects with type ID=11 |
| 326 | 7 | 0 | CECECE | Light gray | Status display "Out of service" mode: fill pattern color | State1FillColor... State7FillColor | Standard view, APL objects with type ID=11 |
| 327 | 7 | 0 | 000000 | Black | Status display "Out of service" mode: font color | State1FontColor... State7FontColor | Standard view, APL objects with type ID=11 |
| 328 | 7 | 0 | 808080 | Dark gray | Status display operating mode: Border color | BorderColor | Standard view, APL objects with type ID=11 |
| 329 | 7 | 0 | FFFFFF | White | Status display control command "Stop"/"Close"/"Off"/"Pause"/"Cancel"/"Requested"/"No": background color | State1BackColor... State7BackColor | Mostly standard/ parameter view/ preview, APL objects with type ID=10, 11, 15 |
| 330 | 7 | 0 | DADADA | Very light gray | Status display control command "Stop"/"Close"/"Off"/"Pause"/"Cancel"/"Requested"/"No": fill pattern color | State1FillColor... State7FillColor | Mostly standard/ parameter view/ preview, APL objects with type ID=10, 11, 15 |
| 331 | 7 | 0 | 000000 | Black | Status display control command "Stop"/"Close"/"Off"/"Pause"/"Cancel"/"Requested"/"No": font color | State1FontColor... State7FontColor | Mostly standard/ parameter view/ preview, APL objects with type ID=10, 11, 15 |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Types views/ objects |
|---|---|---|---|---|---|---|---|
| 332 | 7 | 0 | 00B500 | Bright lime green | Status display control command "Start"/"Open"/"On"/"Continue"/"Active"/"ready"/"Yes": background color | State1BackColor... State7BackColor | Mostly standard/ parameter view/ preview, APL objects with type ID=10, 11, 15, 19 |
| 333 | 7 | 0 | 00FF00 | Pure lime green | Status display control command "Start"/"Open"/"On"/"Continue"/"Active"/"ready"/"Yes": fill pattern color | State1FillColor... State7FillColor | Mostly standard/ parameter view/ preview, APL objects with type ID=10, 11, 15, 19 |
| 334 | 7 | 0 | FFFFFF | White | Status display control command "Start"/"Open"/"On"/"Continue"/"Active"/"ready"/"Yes": font color | State1FontColor... State7FontColor | Mostly standard/ parameter view/ preview, APL objects with type ID=10, 11, 15, 19 |
| 335 | 7 | 0 | FF0000 | Pure red | Status display control command "Rapid stop": background color | State1BackColor... State7BackColor | Standard view, APL objects with type ID=10, 15 |
| 336 | 7 | 0 | FF8296 | Very light red | Status display control command "Rapid stop": fill pattern color | State1FillColor... State7FillColor | Standard view, APL objects with type ID=10, 15 |
| 337 | 7 | 0 | 000000 | Black | Status display control command "Rapid stop": font color | State1FontColor... State7FontColor | Standard view, APL objects with type ID=10, 15 |
| 338 | 7 | 0 | FFFFFF | White | Status display control command "Internal": background color | State1BackColor... State7BackColor | Standard view, APL objects with type ID=11 |
| 339 | 7 | 0 | CECECE | Light gray | Status display control command "Internal": fill pattern color | State1FillColor... State7FillColor | Standard view, APL objects with type ID=11 |
| 340 | 8 | 0 | 000000 | Black | Status display control command "Internal": font color | State1FontColor... State7FontColor | Standard view, APL objects with type ID=11 |
| 341 | 8 | 0 | 0000FF | Pure blue | Status display control command "External": background color | State1BackColor... State7BackColor | Standard view, APL objects with type ID=10, 11, 15 |
| 342 | 8 | 0 | 98BDFF | Very light blue | Status display control command "External": fill pattern color | State1FillColor... State7FillColor | Standard view, APL objects with type ID=10, 11, 15 |
| 343 | 8 | 0 | FFFFFF | White | Status display control command "External": font color | State1FontColor... State7FontColor | Standard view, APL objects with type ID=10, 11, 15 |
| 344 | 8 | 0 | 808080 | Dark gray | Status display control command: Border color | BorderColor | Standard view, APL objects with type ID=10, 11, 15 |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Types views/ objects |
|---|---|---|---|---|---|---|---|
| 345 | 8 | 0 | 000000 | Black | Status display error ("End position error", "Control error", "Invalid Signal", "Changeover error", "Motor protection", "Torque active", "External error", "CPI invalid", "Changeover error", "Fluttering"): Background color | BackColor, State1BackColor...State7BackColor | Standard view, mostly APL objects with type ID=12 |
| 346 | 8 | 0 | 919191 | Dark gray | Status display error ("End position error", "Control error", "Invalid Signal", "Changeover error", "Motor protection", "Torque active", "External error", "CPI invalid", "Changeover error", "Fluttering"): Fill pattern color | FillColor, State1FillColor... State7FillColor | Standard view, mostly APL objects with type ID=12 |
| 347 | 8 | 0 | FFFF00 | Pure yellow | Status display error ("End position error", "Control error", "Invalid Signal", "Changeover error", "Motor protection", "Torque active", "External error", "CPI invalid", "Changeover error", "Fluttering"): Font color | FontColor, State1FontColor...State7FontColor | Standard view, mostly APL objects with type ID=12 |
| 348 | 8 | 0 | 919191 | Dark gray | Status display AS block state ("Forced", "Request 0/1", "Tracking", "Optimizing", "Output ..."[5], "Fuzzy...", "Safety mode", "End", "Taring", "Dribbling", "Paused", "Off"[6], "... Trigger", "Disabled", "SP/MV ramp active"): Background color | BackColor, State1BackColor...State7BackColor | Standard view, mostly APL objects with type ID=12, 13 |
| 349 | 8 | 0 | DADADA | Very light gray | Status display AS block state ("Forced", "Request 0/1", "Tracking", "Optimizing", "Output ..."[7], "Fuzzy...", "Safety mode", "End", "Taring", "Dribbling", "Paused", "Off"[8], "... Trigger", "Disabled", "SP/MV ramp active"): Fill pattern color | FillColor, State1FillColor... State7FillColor | Standard view, mostly APL objects with type ID=12, 13 |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Types views/ objects |
|---|---|---|---|---|---|---|---|
| 350 | 8 | 0 | 000000 | Black | Status display AS block state ("Forced", "Request 0/1", "Tracking", "Optimizing", "Output ..."[9], "Fuzzy...", "Safety mode", "End", "Taring", "Dribbling", "Paused", "Off"[10], "... Trigger", "Disabled", "SP/MV ramp active"): Font color | FontColor, State1FontColor...State7FontColor | Standard view, mostly APL objects with type ID=12, 13 |
| 351 | 8 | 0 | FF0000 | Pure red | Status display alarm ("Flow", "Underdosed", "Overdosed"): Background color | BackColor, State1BackColor...State7BackColor | Standard view, mostly APL objects with type ID=12, 13 |
| 352 | 8 | 0 | FFFFFF | White | Status display alarm ("Flow", "Underdosed", "Overdosed"): Fill pattern color | FillColor, State1FillColor... State7FillColor | Standard view, mostly APL objects with type ID=12, 13 |
| 353 | 8 | 0 | 000000 | Black | Status display alarm ("Flow", "Underdosed", "Overdosed"): Font color | FontColor, State1FontColor...State7FontColor | Standard view, mostly APL objects with type ID=12, 13 |
| 354 | 8 | 0 | 00B500 | Dark lime green | Status display enabled ("On"[11],"Enabled","... dosing"): Background color | BackColor, State1BackColor...State7BackColor | Standard view, mostly APL objects with type ID=12, 13, 15 |
| 355 | 8 | 0 | 00FF00 | Pure lime green | Status display enabled ("On"[12],"Enabled","... dosing"): Fill pattern color | FillColor, State1FillColor... State7FillColor | Standard view, mostly APL objects with type ID=12, 13, 15 |
| 356 | 8 | 0 | FFFFFF | White | Status display enabled ("On"[13],"Enabled","... dosing"): Font color | FontColor, State1FontColor...State7FontColor | Standard view, mostly APL objects with type ID=12, 13, 15 |
| 357 | 8 | 0 | DADADA | Very light gray | Status display disabled ("Off"[14]): Background color | BackColor, State1BackColor...State7BackColor | Standard view, mostly APL objects with type ID=12, 13, 15 |
| 358 | 8 | 0 | FFFFFF | White | Status display disabled ("Off"): Fill pattern color | FillColor, State1FillColor... State7FillColor | Standard view, mostly APL objects with type ID=12, 13, 15 |
| 359 | 8 | 0 | 000000 | Black | Status display disabled ("Off"): Font color | FontColor, State1FontColor...State7FontColor | Standard view, mostly APL objects with type ID=12, 13, 15 |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Types views/ objects |
|---|---|---|---|---|---|---|---|
| 360 | 9 | 0 | FFFA00 | Pure yellow | Status display important AS block information ("Simulation", "Process excitation", "Time delay"): Background color | BackColor, State1BackColor...State7BackColor | Standard/parameter view, mostly APL objects with type ID=12, 13 |
| 361 | 9 | 0 | FFFFFF | White | Status display important AS block information ("Simulation", "Process excitation", "Time delay"): Fill pattern color | FillColor, State1FillColor... State7FillColor | Standard/parameter view, mostly APL objects with type ID=12, 13 |
| 362 | 9 | 0 | 000000 | Black | Status display important AS block information ("Simulation", "Process excitation", "Time delay"): Font color | FontColor, State1FontColor...State7FontColor | Standard/parameter view, mostly APL objects with type ID=12, 13 |
| 363 | 9 | 0 | 0000FF | Pure blue | Status display maintenance ("Maintenance"): Background color | BackColor, State1BackColor...State7BackColor | Standard view, mostly APL objects with type ID=12, 13 |
| 364 | 9 | 0 | 96BBFF | Very light blue | Status display maintenance ("Maintenance"): Fill pattern color | FillColor, State1FillColor... State7FillColor | Standard view, mostly APL objects with type ID=12, 13 |
| 365 | 9 | 0 | 000000 | Black | Status display maintenance ("Maintenance"): Font color | FontColor, State1FontColor...State7FontColor | Standard view, mostly APL objects with type ID=12, 13 |
| 366 | 9 | 0 | 808080 | Dark gray | Status display error, ..., Maintenance: Border color | BorderColor | Standard view / preview, mostly APL objects with type ID=11-13, 15 |
| 367 | 9 | 0 | DADADA | Very light gray | Interlock, display of input values (BOOL): Background color | BackColor | Standard view interlock, APL objects with type ID=100 |
| 368 | 9 | 0 | CECECE | Light gray | Interlock, display of input values (BOOL): Fill pattern color | FillColor | Standard view interlock, APL objects with type ID=100 |
| 369 | 9 | 0 | 000000 | Black | Interlock, display of input values (BOOL): Font color | FontColor | Standard view interlock, APL objects with type ID=100 |
| 370 | 9 | 0 | 808080 | Dark gray | Interlock, display of input values (BOOL): Border color | BorderColor | Standard view interlock, APL objects with type ID=100 |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Types views/ objects |
|---|---|---|---|---|---|---|---|
| 371 | 9 | 0 | 919191 | Dark gray | Interlock, display of input values (BOOL): Background color separator | SeparatorBackColor | Standard view interlock, APL objects with type ID=100 |
| 372 | 9 | 0 | 00FF00 | Pure lime green | Interlock, status "1" for further processing: Background color | SetBackColor | Standard view interlock, APL objects with type ID=101 |
| 373 | 9 | 0 | 00B500 | Dark lime green | Interlock, status "1" for further processing: Fill pattern color | SetFillColor | Standard view interlock, APL objects with type ID=101 |
| 374 | 9 | 0 | FFFFFF | White | Interlock, status "0" for further processing: Background color | ResetBackColor | Standard view interlock, APL objects with type ID=101 |
| 375 | 9 | 0 | B6B6B6 | Gray | Interlock, status "0" for further processing: Fill pattern color | ResetFillColor | Standard view interlock, APL objects with type ID=101 |
| 376 | 9 | 0 | 0000FF | Pure blue | Interlock, status "Bypassed" (excluded) for further processing: Background color | BypassBackColor | Standard view interlock, APL objects with type ID=101 |
| 377 | 9 | 0 | 00A2E8 | Pure blue | Interlock, status "Bypassed" (excluded) for further processing: Fill pattern color | BypassFillColor | Standard view interlock, APL objects with type ID=101 |
| 378 | 9 | 0 | FFFFFF | White | Interlock, status for further processing: Font color | ForeColor | Standard view interlock, APL objects with type ID=101 |
| 379 | 9 | 0 | 6D6D6D | Very dark gray | Interlock, status for further processing: Border color | BorderColor | Standard view interlock, APL objects with type ID=101 |
| 380 | 10 | 0 | B6B6B6 | Gray | Interlock, "Not used" status of the output signal of the interlock block: Background color | Set5BackColor | Standard view interlock, APL objects with type ID=103 |
| 381 | 10 | 0 | DADADA | Very light gray | Interlock, "Not used" status of the output signal of the interlock block: Fill pattern color | Set5FillColor | Standard view interlock, APL objects with type ID=103 |
| 382 | 10 | 0 | 0000FF | Pure blue | Interlock, "Not used" status of the output signal of the interlock block: Background color | Set4BackColor | Standard view interlock, APL objects with type ID=103 |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Types views/ objects |
|---|---|---|---|---|---|---|---|
| 383 | 10 | 0 | 00A2E8 | Pure blue | Interlock, "Not used" status of the output signal of the interlock block: Fill pattern color | Set4FillColor | Standard view interlock, APL objects with type ID=103 |
| 384 | 10 | 0 | E4D400 | Pure yellow | Interlock, "Simulation" status of the output signal of the interlock block: Background color | Set2BackColor | Standard view interlock, APL objects with type ID=103 |
| 385 | 10 | 0 | FFFF00 | Pure yellow | Interlock, "Simulation" status of the output signal of the interlock block: Fill pattern color | Set2FillColor | Standard view interlock, APL objects with type ID=103 |
| 386 | 10 | 0 | 00B500 | Bright lime green | Interlock, "Unlocked" status of the output signal of the interlock block: Background color | Set1BackColor | Standard view interlock, APL objects with type ID=103 |
| 387 | 10 | 0 | 00FF00 | Pure lime green | Interlock, "Unlocked" status of the output signal of the interlock block: Fill pattern color | Set1FillColor | Standard view interlock, APL objects with type ID=103 |
| 388 | 10 | 0 | E60000 | Pure red | Interlock, "Locked" status of the output signal of the interlock block: Background color | Set3BackColor | Standard view interlock, APL objects with type ID=103 |
| 389 | 10 | 0 | FF0000 | Pure red | Interlock, "Locked" status of the output signal of the interlock block: Fill pattern color | Set3FillColor | Standard view interlock, APL objects with type ID=103 |
| 390 | 10 | 0 | 000000 | Black | Interlock, status of the output signal of the interlock block: Font color | ForeColor | Standard view interlock, APL objects with type ID=103 |
| 391 | 10 | 0 | 919191 | Dark gray | Interlock, status of the output signal of the interlock block: Border color | BorderColor | Standard view interlock, APL objects with type ID=103 |
| 392 | 10 | 0 | 850082 | Dark magenta | Interlock, "First signal" status display: Background color | BackColor | Standard view interlock, APL objects with type ID=102 |
| 393 | 10 | 0 | CD51FF | Light purple | Interlock, "First signal" status display: Fill pattern color | FillColor | Standard view interlock, APL objects with type ID=102 |
| 394 | 10 | 0 | 808080 | Dark gray | Interlock, "First signal" status display: Border color | BorderColor | Standard view interlock, APL objects with type ID=102 |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Types views/ objects |
|---|---|---|---|---|---|---|---|
| 395 | 10 | 0 | 00B500 | Bright lime green | Interlock, "Unlocked" status of the block output: Line color | OnStateColor | Standard view interlock, APL objects with type ID=104 |
| 396 | 10 | 0 | FFFFFF | White | Interlock, "Locked" status of the block output: Line color | OffStateColor | Standard view interlock, APL objects with type ID=104 |
| 397 | 10 | 0 | B6B6B6 | Gray | Interlock, block rectangle: Background color | BackgroundColor | Standard view interlock, HMIRectangle |
| 398 | 10 | 0 | DADADA | Very light gray | Interlock, block rectangle: Fill pattern color | FillPatternColor | Standard view interlock, HMIRectangle |
| 399 | 10 | 0 | 919191 | Dark gray | Interlock, block rectangle: Border color | BorderColor | Standard view interlock, HMIRectangle |
| 400 | 11 | 0 | 00B500 | Bright lime green | Bar chart, analog process value: Background color | ValueColor | Standard/setpoint view, APL objects with type ID=71 |
| 401 | 11 | 0 | 00FF00 | Pure lime green | Bar chart, analog process value: Fill pattern color | ValueFillColor | Standard/setpoint view, APL objects with type ID=71 |
| 402 | 11 | 0 | 0000FF | Pure blue | Bar chart, analog setpoint: Background color, foreground color | ValueColor, FontColor | Standard/setpoint view, APL objects with type ID=70, 80, 81 |
| 403 | 11 | 0 | 98BDFF | Very light blue | Bar chart, analog setpoint: Fill pattern color | ValueFillColor | Standard/setpoint view, APL objects with type ID=70 |
| 404 | 11 | 0 | D27A00 | Bright orange | Bar chart, changed analog variable: Background color, foreground color | ValueColor, FontColor | Standard view, APL objects with type ID=72, 82, 83 |
| 405 | 11 | 0 | FFC848 | Light orange | Bar chart, changed analog variable: Fill pattern color | ValueFillColor | Standard view, APL objects with type ID=72 |
| 406 | 11 | 0 | 008582 | Dark cyan | Bar chart, analog readback value: Background color | ValueColor | Standard view, APL objects with type ID=71, 73 |
| 407 | 11 | 0 | 00CECA | Bright cyan | Bar chart, analog readback value: Fill pattern color | ValueFillColor | Standard view, APL objects with type ID=71, 73 |
| 408 | 11 | 0 | 008582 | Dark cyan | Bar chart, predicted value: Background color | ValueColor | Standard view, APL objects with type ID=75 |
| 409 | 11 | 0 | 00CECA | Bright cyan | Bar chart, predicted value: Fill pattern color | ValueFillColor | Standard view, APL objects with type ID=75 |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Types views/ objects |
|---|---|---|---|---|---|---|---|
| 410 | 11 | 0 | 000000 | Black | Bar chart, message gradient value or limit (without alarm): Background color | ValueColor, AlarmColor, WarningColor | Standard view, APL objects with type ID=74 or 71[15] |
| 411 | 11 | 0 | 6D6D6D | Very dark gray | Bar chart, gradient value: Fill pattern color | ValueFillColor | Standard view, APL objects with type ID=74 |
| 412 | 11 | 0 | 808080 | Dark gray | Bar chart: Border color | BorderColor | Standard/setpoint view, APL objects with type ID=7075 |
| 413 | 11 | 0 | 62ACAA | Mostly unsaturated dark cyan | Selection rectangle operation: Background color | BackgroundColor | All views with operation, HMIRectangle object name "rect_selectionBorder" |
| 414 | 11 | 0 | B1CFCD | Gray cyan | Selection rectangle operation: Fill pattern color | FillColor | All views with operation, HMIRectangle object name "rect_selectionBorder" |
| 415 | 11 | 0 | 808080 | Dark gray | Selection rectangle operation: Border color | BorderColor | All views with operation, HMIRectangle object name "rect_selectionBorder" |
| 416 | 11 | 0 | 808080 | Dark gray | Dividing line: Border color | BorderColor | All views with dividing line, HMILine |

[1] Note: Also types without messages have an empty group display

[2] Note: Only important for the "Next" button

[3] "Lock"/"Acknowledgement"/"Attach"/"Back to process picture" buttons; "Attach" button in the operating area

[4] In the standard view Ratio

[5] Used in type ID = 17, 18, 120

[6] In the standard view DoseL

[7] In the standard view Ratio

[8] In the standard view DoseL

[9] In the standard view Ratio

[10] In the standard view DoseL

[11] In the standard view OpDi01, OpDi03, display the confirmation of the command

[12] In the standard view OpDi01, OpDi03, display the confirmation of the command

[13] Not used, only in @PCS7ElementsAPL.pdl

[14] In the standard view OpDi01, OpDi03, display the confirmation of the command

[15] In @PG_CntOhSc_Standard.PDL

## Color indexes for trends

The following color indexes can be created for trends:

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Type block icons |
|---|---|---|---|---|---|---|---|
| 417 | 11 | 0 | 00CE00 | Bright lime green | Actual value | TrendColor1 - TrendColor12 | Trend view |
| 418 | 11 | 0 | 0000FF | Pure blue | Setpoint | TrendColor1 - TrendColor12 | Trend view |
| 419 | 11 | 0 | E69100 | Pure orange | Changed variable | TrendColor1 - TrendColor12 | Trend view |
| 420 | 12 | 0 | 008582 | Dark cyan | Read-back value | TrendColor1 - 1TrendColor12 | Trend view |
| 421 | 12 | 0 | 00CE00 | Bright lime green | Controller, closed loop | TrendColor8 | Trend view controller |
| 422 | 12 | 0 | 000000 | Black | Controller, input for control performance index (CPI) | TrendColor7 | Trend view controller |
| 423 | 12 | 0 | 000000 | Black | Controller, binary message "violated"[1] | TrendColor3, TrendColor4, TrendColor9, TrendColor10 | Trend view controller |
| 424 | 12 | 0 | B100B5 | Bright magenta | Controller, control performance index (CPI) | TrendColor1 | Trend view ConPerMon |
| 425 | 12 | 0 | E69100 | Pure orange | Ratio, analog input "In" | TrendColor1 | Trend view Ratio |
| 426 | 12 | 0 | 00FF00 | Pure lime green | Ratio, input for process variable "InPV" | TrendColor2 | Trend view Ratio |
| 427 | 12 | 0 | 0000FF | Pure blue | Ratio, process value of the secondary component "SecComPV" | TrendColor3 | Trend view Ratio |
| 428 | 12 | 0 | B100B5 | Bright magenta | Ratio, output "Out" | TrendColor4 | Trend view Ratio |
| 429 | 12 | 0 | FF0000 | Pure red | Ratio, current ratio "RatioPV" | TrendColor5 | Trend view Ratio |
| 430 | 12 | 0 | 6D6D6D | Very dark gray | Ratio, applied ratio (RatioInt or RatioExt) "RatioOut" | TrendColor6 | Trend view Ratio |
| 431 | 12 | 0 | 009D00 | Dark lime green | Dosing, dosing amount | TrendColor3 | Trend view Dosing |
| 432 | 12 | 0 | FF0000 | Pure red | Monitoring, gradient value | TrendColor2 | Trend view MonAnL |
| 433 | 12 | 0 | E69100 | Pure orange | Monitoring, output 1 | TrendColor1 | Trend view MonDi08, MonDiL, MonDiS |
| 434 | 12 | 0 | 00CE00 | Bright lime green | Monitoring, output 2 | TrendColor2 | Trend view MonDi08 |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Type block icons |
|---|---|---|---|---|---|---|---|
| 435 | 12 | 0 | 0000FF | Pure blue | Monitoring, output 3 | TrendColor3 | Trend view MonDi08 |
| 436 | 12 | 0 | B100B5 | Bright magenta | Monitoring, output 4 | TrendColor4 | Trend view MonDi08 |
| 437 | 12 | 0 | FF0000 | Pure red | Monitoring, output 5 | TrendColor5 | Trend view MonDi08 |
| 438 | 12 | 0 | 494949 | Very dark gray | Monitoring, output 6 | TrendColor6 | Trend view MonDi08 |
| 439 | 12 | 0 | 543500 | Very dark orange | Monitoring, output 7 | TrendColor7 | Trend view MonDi08 |
| 440 | 13 | 0 | 005400 | Very dark lime green | Monitoring, output 8 | TrendColor8 | Trend view MonDi08 |
| 441 | 13 | 0 | 00CE00 | Bright lime green | Monitoring, flutter value | TrendColor2 | Trend view MonDiL |
| 442 | 13 | 0 | E69100 | Pure orange | Operation, binary output / binary output 1 | TrendColor1 | Operation, trend view binary |
| 443 | 13 | 0 | 494949 | Very dark gray | Operation, binary output 2 | TrendColor2 | Operation, trend view binary |
| 444 | 13 | 0 | 0000FF | Pure blue | Operation, binary output 3 | TrendColor3 | Operation, trend view binary |
| 445 | 13 | 0 | B100B5 | Bright magenta | Operation, binary feedback 1 | TrendColor4 | Operation, trend view binary |
| 446 | 13 | 0 | FF0000 | Pure red | Operation, binary feedback 2 | TrendColor5 | Operation, trend view binary |
| 447 | 13 | 0 | 00CE00 | Bright lime green | Operation, binary feedback / binary feedback 3 | TrendColor2, TrendColor6 | Operation, trend view binary |
| 448 | 13 | 0 | 0000FF | Pure blue | Actuators, valve control output | TrendColor1 - TrendColor4, TrendColor6 | Trend view Valve |
| 449 | 13 | 0 | 000000 | Black | Actuators, valve feedback open | TrendColor1 | Trend view Valve |
| 450 | 13 | 0 | 6D6D6D | Very dark gray | Actuators, valve feedback close | TrendColor2 | Trend view Valve |
| 451 | 13 | 0 | 005400 | Very dark lime green | Actuators, valve feedback | TrendColor1, TrendColor3, TrendColor5, TrendColor7 | Trend view Vlv2WayL |
| 452 | 13 | 0 | E69100 | Pure orange | Actuators, open valve output | TrendColor1 | Trend view VlvMotL |
| 453 | 13 | 0 | 00CE00 | Bright lime green | Actuators, close valve output | TrendColor2 | Trend view VlvMotL |
| 454 | 13 | 0 | 0000FF | Pure blue | Actuators, valve feedback open | TrendColor3 | Trend view VlvMotL |
| 455 | 13 | 0 | B100B5 | Bright magenta | Actuators, valve feedback close | TrendColor4 | Trend view VlvMotL |
| 456 | 13 | 0 | FF0000 | Pure red | Actuators, valve feedback open | TrendColor5 | Trend view VlvMotL |
| 457 | 13 | 0 | 005400 | Very dark lime green | Actuators, valve feedback close | TrendColor6 | Trend view VlvMotL |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Type block icons |
|---|---|---|---|---|---|---|---|
| 458 | 13 | 0 | E69100 | Pure orange | Actuators, motor start output | TrendColor1 | Trend view MotL/MotS |
| 459 | 13 | 0 | 00CE00 | Bright lime green | Actuators, motor feedback | TrendColor2 | Trend view MotL/MotS |
| 460 | 14 | 0 | E69100 | Pure orange | Drives, motor start Speed1 / forward | TrendColor1 | Trend view MotSpdL/MotRevL/ MotSpdCL |
| 461 | 14 | 0 | 00CE00 | Bright lime green | Drives, motor start Speed2 / reverse | TrendColor2 | Trend view MotSpdL/MotRevL/ MotSpdCL |
| 462 | 14 | 0 | 0000FF | Pure blue | Drives, feedback Speed1 / forward | TrendColor3 | Trend view MotSpdL/MotRevL |
| 463 | 14 | 0 | B100B5 | Bright magenta | Drives, feedback Speed2 / reverse | TrendColor4 | Trend view MotSpdL/MotRevL |
| 464 | 14 | 0 | 00FFFA | Pure cyan | Drives, feedback forward | TrendColor3 | Trend view MotSpdCL |
| 465 | 14 | 0 | B100B5 | Bright magenta | Drives, feedback reverse | TrendColor4 | Trend view MotSpdCL |
| 466 | 14 | 0 | E69100 | Pure orange | Count: Output | TrendColor1 | Trend view Counter |
| 467 | 14 | 0 | E69100 | Pure orange | Count / time: Unit on | TrendColor1 | CountOh / TimeTrig |
| 468 | 14 | 0 | 005400 | Very dark lime green | Count / time: Count type | TrendColor7 | CountOh / TimeTrig |
| 469 | 14 | 0 | E69100 | Pure orange | Count / time: Count type | TrendColor1 | Trend view CntOhSc |
| 470 | 14 | 0 | 00CE00 | Bright lime green | Count / time: Unit on | TrendColor2 | Trend view CntOhSc |
| 471 | 14 | 0 | 005400 | Very dark lime green | Count: Service life | TrendColor7 | Trend view CntOhSc |
| 472 | 14 | 0 | 6D6D6D | Very dark gray | Count: Count | TrendColor8 | Trend view CntOhSc |
| 473 | 14 | 0 | 0000FF | Pure blue | Statusdisplay operation command - Release for Maintenance" ("Yes"): background color | State1BackColor | Mostly In Parameter view APL objects with TypeID=11 |
| 474 | 14 | 0 | FFFFFF | White | Statusdisplay operation command - Release for Maintenance" ("Yes"): filling pattern color | State1FillColor | Mostly In Parameter view APL objects with TypeID=11 |
| 475 | 14 | 0 | FFFFFF | White | Statusdisplay operation command -Release for Maintenance ("Yes"): font color | State1FontColor | Mostly In Parameter view APL objects with TypeID=11 |
| 476 | 14 | 0 | 919191 | Dark gray | Statusdisplay ("Not Ready", "None", "No"): background color | State1BackColor... State7BackColor | Mostly In ShrdResS Standard \Preview view APL objects with TypeID=11,19 |

| Color index | Section | Transparency | HTML code (RGB) | Color name | Description | Property(ies) [attribute name] | Type block icons |
|---|---|---|---|---|---|---|---|
| 477 | 14 | 0 | DADADA | Very light gray | Statusdisplay (“Not Ready”, “None”,”No”): filling pattern color | State1FillColor... State7FillColor | Mostly In ShrdResS Standard \Preview view APL objects with TypeID=11,19 |
| 478 | 14 | 0 | 000000 | Black | Statusdisplay (“Not Ready”, “None”,”No”): font color | State1FontColor... State7FontColor | Mostly In ShrdResS Standard \Preview view APL objects with TypeID=11,19 |
| 479 | 14 | 0 | 757575 | Gray | Analog value in views for high and low scale. | FontColor | Mostly In Standard \Limit view I/O fields (also extended) |

[1] For example, PIDConL "PV_ToleHi#Value", ".MV_HiAct#Value"

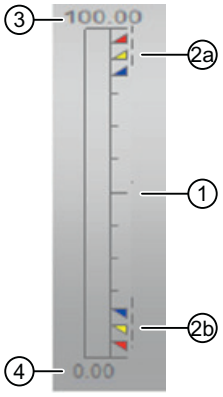
[2] ModPreCon=@PG_APL_TrendMPC.PDL; MPC10x10=@PG_APL_TrendMPC_L.PDL

**Note**

If these colors were changed only via the OS project editor, the default values can be assigned again.

## 2.3.29 Bar graph

### Bar graph for the Process Value




(1) Bar graph for the "Process Value"

This area shows the current "Process value" in the form of a bar graph. You can also see the high low limits of the bar graph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

**Note**

The value displayed can be different for different blocks. For example, the Bar graph display other than "Process Value" are as follows

Bar graph display other than "Process Value" are as follows:

| Block name | Value and Limits displayed | Color |
|---|---|---|
| DoseL | Current quantity | |
| CntOhSc | Current counts (Cnt and Time-Hours(h)). | |
| CountOh | Current count (TimeMin (min)). | |
| CountScL | Current count (Out). | |
| TotalL | Current sum value (Out). | Orange |
| TimeTrig | Amount of time(s) until the next trigger | |
| ConPerMon | Current control performance index (CPI) | |
| MotL, MotRevL, MotSpdCL, MotSpdL, VlvMotL, VlvPosL | Current additional value (AV_Out) in the limit view for additional value. | |
| MotSpdCL | Current readback value in the standard view | Cyan |
| PumpMon | Electrical power<br>Mechanical power<br>Hydraulic power | Green<br>Yellow<br>Cyan |

(2) Limit display

- 2a: High limit display
- 2b: Low limit display

These colored triangles indicate the configured limits in the respective bar graph:

- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

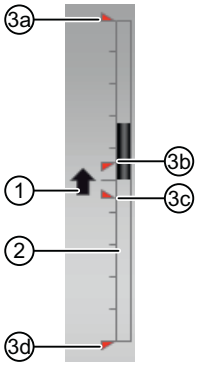
**Note**

- The symbols displayed are not valid for user-configured message classes. Please take into consideration the validity of terms for User-configured message classes.
- The limit triangles visibility depends on the configuration in engineering system (ES).
  - If a limit is not enabled (e.g. `PV_AH_En` = 0 & `Feature.Bit23` =0) then triangle for corresponding limit (high alarm in this example) will not be visible.
  - If a limit is not supported by a block, then the corresponding limit triangle is not displayed. E.g. `DoseL` block only has Alarm high and Alarm low limits. So, limit triangles for Warning and Tolerence will not be visible.
  - For `TotalL`, `CountOh` and `CountScL`, in counting mode "count up", only the colored triangles of the upper limits are visible; and in counting mode "count down", only the colored triangles of the lower limits are visible.
  - For `MotSpdCL`, only the warning high and low limit triangles for readback value are displayed.
  - For `CntOhSc`, only the high limit triangles are available for the following configured limits in the respective bar graph:
    Red: High limit 1
    Yellow: High limit 2
  - For `ConPerMon`, only the low limit triangles are available for the following configured limits in the respective bar graph:
    Red: Low alarm limit
    Yellow: Low warning limit
    For `PumpMon`, only the high and low limit triangles for Alarms and warnings for readback value are displayed.

(3) Bargraph high scale

(4) Bargraph low scale

### Bar graph for the "Gradient value"




(1) Display of the gradient indicator

This display indicates the movement of the gradient up (↑) or down (↓).

Gradient monitoring is displayed when the gradient value PV_Grad ≠ 0 and one of the following monitoring functions is activated:

- Gradient monitoring for positive changes (`GradHUpEn` = 1)
- Gradient monitoring for negative changes (`GradHDnEn` = 1)
- Gradient monitoring (`GradLEn` = 1)

(2) Bar graph for the gradient

This area shows the current gradient value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

The bar graph is displayed when gradient monitoring is activated for positive (`GradHUpEn` = 1) or negative changes (`GradHDnEn` = 1).

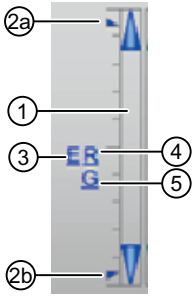
(3) Display of limits in the bar graph

This area shows you the specified limits.

The display only appears when the bar for the gradients is also displayed.

- 3a: Limit (high) for positive gradients (`GradHUpLim`)
- 3b, 3c: Limit (low) for positive and negative gradients (`GradLLim`)
- 3d: Limit (high) for negative gradients (`GradHDnLim`)

### Bar graph for the "Setpoint value"




(1) Bar graph for the setpoint

This area shows the current setpoint in the form of a bar graph. You can also see the high low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

**Note**

For Ratio block, this area shows you the current value for the ratio (`RatioOut`).

(2) Limit display for the setpoint

These triangles show the setpoint limits configured in the ES.

- 2a: High limit for setpoint
- 2b: low limit for setpoint

**Note**

For Ratio block, these blue triangles show the configured High and low range limits for the ratio.

(3) Display of external setpoint

This display [E] is only visible when you have selected "Internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

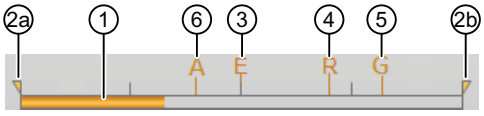
(4) Display for the target setpoint of the setpoint ramp

This display [R] shows the target setpoint of the setpoint ramp and is only visible if you have enabled ramp generation in the Ramp view (Page 334).

(5) Display for the target setpoint of the setpoint gradient ramp

This display [G] shows the target setpoint of the setpoint ramp and is only visible if you have enabled gradient ramp generation in the Ramp view (Page 334).

### Bar graph for the "Manipulated value"




(1) Bar graph for the manipulated variable

This area shows the current manipulated variable in the form of a bar graph. You can also see the high and low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

**Note**

For Ratio block this display graphically represents the output value, `Out`

(2) Limit display for the manipulated variable

These orange triangles show the limits configured in the ES.

- 2a: low limit for manipulated variable
- 2b: High limit for manipulated variable

**Note**

For Ratio block limits are for output value `Out`.

(3) Display of external manipulated variable

This display [E] is only visible when you have selected "Internal" manipulated variable specification. It shows the external manipulated variable that would apply if you were to change the manipulated variable specification to "External".

(4) Display for the target manipulated variable of the manipulated variable ramp

This display [R] shows the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 334).

(5) Display for the target manipulated variable of the manipulated variable gradient ramp

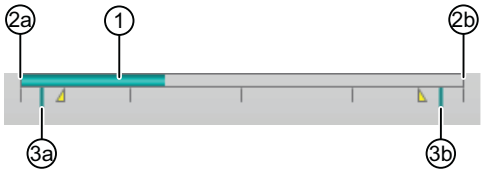
This display [G] shows the target setpoint and is only visible if you have enabled gradient ramp generation in the Ramp view (Page 334).

(6) Display of the manipulated variable for automatic mode

The display [A] is only visible when the block is not in automatic mode. In Automatic mode, it is displayed when it needs to be reset or when tracking or forced tracking is enabled.

This display shows the manipulated variable that would apply if you were to switch to automatic mode or if you were to reset in automatic mode or disable tracking or forced tracking.

### Bar graph for the "Position Feedback”




(1) Bar graph for position feedback

This display is only visible when the corresponding block input is connected.

This area shows the current position feedback in the form of a bar graph. You can also see the high and low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(2) Limit display for the position feedback

These triangles show the limits configured in the ES.

- 2a. low limit for position feedback
- 2b. High limit for position feedback

The limit triangles can have different colors based on the type of limit:

- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

**Note**

**The limit triangles visibility depends on the configuration in engineering system (ES).**

If a limit is not enabled (e.g. `RbkWH_En`= 0 & `Feature.Bit31` =0 for `PIDConL`) then triangle for corresponding limit (Warning high limit in this example) will not be visible.

(3) The limits for the "open" and "close" positions (for `VlvAnL`)

- 3a: limits for the "close" position
- 3b: limits for the "open" position

## 2.3.30 Reset of errors during Request 0/1

"Request 0/1" is displayed in faceplate whenever there is a reset pending but reset from faceplate is not possible.

The following table shows the various possibilities to reset the errors.

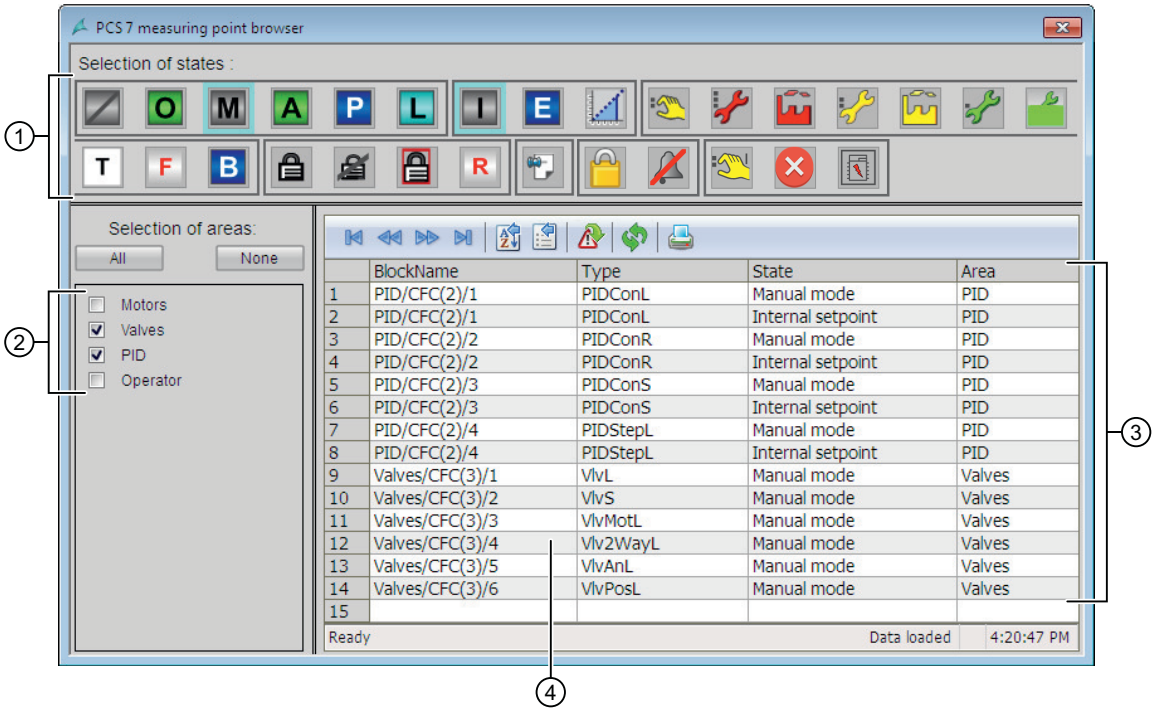
| Mode | `OS_Perm.Bit7` | `Feature.Bit30` | `Feature.Bit31` | `Feature.Bit9` | `Feature2.Bit13` | Resetting of errors |
|---|---|---|---|---|---|---|
| Auto | 0 | 0 | 0 | 0 | 0 | Possible by `RstLi.Value` |
| | 1 | 1 | 0 | 0 | 0 | |
| | 0 | 0 | 0 | 1 | 0 | Possible by two ways<br><br>1) `RstLi.Value` or<br>2) By Auto Commands |
| | 1 | 1 | 0 | 1 | 0 | |
| Manual | 0 | 1 | 1 | 0 | 0 | Possible by forced commands |
| | 0 | 0 | 1 | 0 | 0 | Possible by `RstLi.Value` |
| Local | 0 | 0 | 0 | 0 | 1 | Reset is possible only via Faceplate hence not recommended to Remove the `OS_Perm.Bit7` |

# 2.4 PCS 7 measuring point browser

## 2.4.1 Overview of the "PCS 7 measuring point browser" window

### Overview

The "PCS 7 measuring point browser" window displays all relevant status information of the APL function blocks.




### (1) "Selection of states"

This area displays all possible states of APL function blocks. All state icons are shown in their respective status group frames. The group name is displayed in the tooltip when the mouse pointer is hovered in the area between the state icon and the group frame. You can select/ deselect one or more states from any status group by clicking the icon. The status information of the selected states is displayed in the status area.

The following states are displayed in this area:

| Icon | State | Status group |
|---|---|---|
|  | Out of service | Operating mode |
|  | On mode | |
|  | Manual mode | |
|  | Automatic mode | |
|  | Program mode | |
|  | Local mode | |
|  | Internal setpoint | Setpoint |
|  | External setpoint | |
|  | Ramp active | |
|  | Manipulated value | Signal status |
|  | Bad, device-related | |
|  | Bad, process-related | |
|  | Uncertain, device-related | |
|  | Uncertain, process-related | |
|  | Maintenance request | |
|  | In progress | |
|  | Tracking | Tracking, Forcing and Bypass |
|  | Forcing | |
|  | Bypass | |

| Icon | State | Status group |
|---|---|---|
|  | Interlock active | Interlock |
|  | Interlock disabled | |
|  | Rapid stop | |
|  | Reset request | |
|  | Active memo | Memo |
|  | Locked messages | Messages |
|  | Suppressed messages | |
|  | Internal simulation | Block status |
|  | Internal error | |
|  | Group error | |

### (2) "Selection of areas"

This area displays the top folders of the plant hierarchy. You can select/deselect one or more areas. The status information from the selected areas is displayed in the status area.

### (3) Status area

This area shows the list of blocks with their corresponding "Type", "State", and "Area" depending on the "Selection of states" and "Selection of areas".

### (4) Opening the faceplate

Double-click a row in the status area to open the corresponding faceplate.

# Operator control blocks

# 3

## 3.1 Comparison of large & small blocks

### 3.1.1 OpAnL compared to OpAnS

#### Comparison of the OpAnL and OpAnS blocks

The following tables are intended to help you decide which block to use.

#### Memory and runtime savings of the small block compared to the large block

You save the following resources for each instance:

- Memory space: ~ 60%
- Runtime: ~ 30%

#### Block operating modes

| | OpAnL | OpAnS |
|---|---|---|
| On (Page 77) | X | X |
| Out of service (Page 77) | X | X |

#### Functions of the blocks

| | OpAnL | OpAnS |
|---|---|---|
| Setpoint specification - internal/ external (Page 136) | X | X |
| Setpoint limiting for external set-points (Page 222) | X | |
| Using setpoint ramp (Page 131) | X | |
| Gradient limit of the setpoint (Page 132) | X | |
| Forming and outputting the signal status for technologic blocks (Page 117) | X | X |
| Selecting a unit of measure (Page 240) | X | X |
| Simulating signals (Page 64) | X | X |
| Operator control permissions (Page 285) | X | X |

| | OpAnL | OpAnS |
|---|---|---|
| Display and operator input area for process values and setpoints (Page 233) | X | |
| Opening additional faceplates (Page 234) | X | X |
| SIMATIC BATCH functionality (Page 73) | X | X |
| Generating instance-specific messages (Page 230) | X | |
| Suppressing messages using the MsgLock parameter (Page 231) | X | |
| Process value with separate scale range (Page 173) | X | |

## Configurable functions using the `Feature` parameter

| Bit number | `Feature` bit function | OpAnL | OpAnS |
|---|---|---|---|
| 0 | Set startup characteristics (Page 148) | X | X |
| 1 | Reaction to the out of service mode (Page 188) | X | X |
| 16 | Process value with separate scale range (Page 173) | X | |
| 22 | Update acknowledgment and error status of the message call (Page 171) | X | |
| 24 | Enabling local operator authorization (Page 169) | X | X |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) | X | X |

## See also

Manual and automatic mode for control blocks (Page 79)

# 3.2 OpAnL - Check and output analog signals (large)

## 3.2.1 Description of OpAnL

### Object name (type + number) and family

Type + number: FB 1865

Family: Operate

### Area of application for OpAnL

The block is used for the following applications:

- Checking and transferring analog input values

### How it works

The block checks incoming, internal (entered in the faceplate) or external (CFC/SFC) analog signals for their limits at the `SP_Int` or `SP_Ext` input and forwards them to the output `SP`, depending on the setting of the `SP_LiOp` input parameter.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

### Status word allocation for `Status1` parameter

For a description of the individual parameters, see the section OpAnL I/Os (Page 397).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | Not used |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | Invalid signal status |
| 6 | `OnAct.Value` |
| 7 | `SP_UpRaAct`, `SP_DnRaAct` limits enabled for gradient mode (`SP_RateOn = 1`) |

| Status bit | Parameter |
|---|---|
| 8 | SP_ExtAct.Value |
| 9 | SP_LoAct.Value |
| 10 | SP_HiAct.Value |
| 11 | Resynchronization: Display preview |
| 12 - 13 | Not used |
| 14 | SP_RmpOn |
| 15 | SP_RmpModTime |
| 16 - 17 | Not used |
| 18 | Feature.Bit16 |
| 19 - 21 | Not used |
| 22 | 1 = SP ramp active |
| 23 - 28 | Not used |
| 29 | 1 = Input parameter SP_LiRmpTime is not interconnected (SP_LiRmpTime.ST = 16#FF) |
| 30 | Internal error |
| 31 | MsgLock |

**See also**

OpAnL functions (Page 391)

OpAnL messaging (Page 395)

OpAnL block diagram (Page 403)

OpAnL error handling (Page 394)

OpAnL modes (Page 390)

## 3.2.2 OpAnL modes

### OpAnL modes

The block can be operated using the following modes:

- On (Page 77)
- Out of service (Page 77)

### "On"

General information on the "On" mode is available in the section On (Page 77).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

**See also**

OpAnL I/Os (Page 397)
OpAnL messaging (Page 395)
OpAnL error handling (Page 394)
Description of OpAnL (Page 389)
OpAnL functions (Page 391)
OpAnL block diagram (Page 403)

## 3.2.3 OpAnL functions

### Functions of OpAnL

The functions for this block are listed below.

### Suppressing messages using the MsgLock parameter

This block provides the standard function Resynchronization (Page 258)

### Internal or external setpoint selection

This block provides the standard function Setpoint specification - internal/external (Page 136).

### Setpoint limitation

Use the `SP_HiLim` and `SP_LoLim` input parameters to limit the setpoint to maximum and minimum limits. If a limit is violated, the setpoint is limited to the limits you have set. If the limits are infringed, the output parameters `SP_HiAct` and `SP_LoAct` display 1.

### Using setpoint ramp

This block provides the standard function Using setpoint ramp (Page 131).

### Gradient limit of the setpoint

This block provides the standard function Gradient limit of the setpoint (Page 132).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `SP_Out.ST`
- `PV_In.ST`
- `SP_Ext.ST` (only if `Feature.Bit10` = 1)

The signal status of the `SP_Out` output parameter is always equivalent to the signal status of input parameter `SP_Ext` or `SP_Int`, depending on how the setpoint is specified. If the internal setpoint `SP_Int` is used, the signal status is always output as 16#80.

In case of `Feature.Bit10` = 1, `SP_Ext.ST` influences `ST_Worst` independent of setpoint specification.

If an external setpoint is used, the signal status 16#60 (external simulation) is suppressed because the block acts as a sink with external simulation.

### Considering bad quality of automatic commands or external values

If the `Feature` bit Considering bad quality of automatic commands or external values (Page 200) is set to 1 and the parameter `SP_Ext` has bad signal status (16#00 or 16#28), the block works with the last valid value of `SP_Ext` in the "On" mode.

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

### Resynchronization

This block provides the standard function Resynchronization (Page 258).

Resynchronization is switched on over feature bits:

| | |
|---|---|
| Feature Bit17 =1 | Resynchronization after CPU startup |
| Feature Bit18 =1 | Resynchronization after Out of service |
| Feature Bit20 =1 | Resynchronization with 0->1 trigger at StateTrkOn input |

If there are no restrictions the control outputs are tracked as follow:

SP_Out := PV_In;

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 6 | Ramp rate calculation (Page 191) |
| 10 | Considering bad quality of automatic commands or external values (Page 200) |
| 12 | Gradient limitation with time duration (Page 197) |
| 16 | Process value with separate scale range (Page 173) |
| 17 | Resynchronization after startup (Page 213) |
| 18 | Resynchronization after Out of service (Page 213) |
| 19 | Resynchronization after CSF (Page 214) |
| 20 | Resynchronization with StateTrkOn input (Page 214) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 24 | Enabling local operator authorization (Page 169) |

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 - 9 | Not used |
| 10 | 1 = Operator can switch to internal |
| 11 | 1 = Operator can switch to external |
| 12 | 1 = Operator can enable bumpless switchover |
| 13 | 1 = Operator can change `SP_Int` |
| 14 | 1 = Operator can enable `SP_RateOn` |
| 15 | 1 = Operator can change `SP_UpRaLim` |
| 16 | 1 = Operator can change `SP_DnRaLim` |
| 17 | 1 = Operator can activate the setpoint ramp (`SP_RmpOn`) |
| 18 | 1 = Operator can switch between specification of the duration (`SP_RmpTime`) and gradient (`SP_DnRaLim`, `SP_UpRaLim`) for calculating the ramp slope |
| 19 | 1 = Operator can change the time for the setpoint ramp (`SP_RmpTime`) |
| 20 | 1 = Operator can change the target setpoint (`SP_RmpTarget`) |
| 21 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Specifying the display area for process and setpoint values as well as operations

The block provides the standard function Display and operator input area for process values and setpoints (Page 233).

### Opening additional faceplates

The block provides the standard function Opening additional faceplates (Page 234).

### SIMATIC BATCH functionality

The block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of OpAnL (Page 389)

OpAnL messaging (Page 395)

OpAnL I/Os (Page 397)

OpAnL block diagram (Page 403)

OpAnL error handling (Page 394)

OpAnL modes (Page 390)

Suppressing messages using the MsgLock parameter (Page 231)

## 3.2.4 OpAnL error handling

### Error handling of OpAnL

Refer to the section Error handling (Page 128) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 51 | Invalid signal<br>`SP_LiOp` = 1 and `SP_ExtLi` = 1 and `SP_IntLi` = 1 |

### See also

OpAnL block diagram (Page 403)
OpAnL I/Os (Page 397)
OpAnL messaging (Page 395)
OpAnL functions (Page 391)
OpAnL modes (Page 390)
Description of OpAnL (Page 389)

## 3.2.5 OpAnL messaging

### Messaging

The following messages can be generated for this block:

- Process messages

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId | SIG 1 | AS process control message - fault | $$BlockComment$$<br>External message 1 |
| | SIG 2 | AS process control message - fault | $$BlockComment$$<br>External message 2 |
| | SIG 3 | AS process control message - fault | $$BlockComment$$<br>External message 3 |
| | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External message 4 |
| | SIG 5 | Reserved | $$BlockComment$$<br>Reserved |
| | SIG 6 | Reserved | $$BlockComment$$<br>Reserved |
| | SIG 7 | Reserved | $$BlockComment$$<br>Reserved |
| | SIG 8 | Reserved | $$BlockComment$$<br>Reserved |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance **MsgEvId**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |
| 4 | ExtVa104 |
| 5 | ExtVa105 |
| 6 | ExtVa106 |
| 7 | ExtVa107 |

The associated values 4 ... 7 are allocated to the parameters ExtVa104 ... ExtVa107 and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

OpAnL error handling (Page 394)

OpAnL modes (Page 390)

OpAnL block diagram (Page 403)

## 3.2.6 OpAnL I/Os

### OpAnL I/Os

For the parameters that are not linkable refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkTypeData* | This input can not be edit or linked and is for internal usage only! | | |
| EN | 1 = Called block will be processed | BOOL | 1 |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg3 | Binary input for freely selectable message 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg4 | Binary input for freely selectable message 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ExtVa104 | Associated value 4 for messages (MsgEvID1) | ANY | |
| ExtVa105 | Associated value 5 for messages (MsgEvID1) | ANY | |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| Feature | I/O for additional functions (Page 391) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| MsgEvId | Message number (assigned automatically) | DWORD | 16#00000000 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operating permission, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operating permissions (Page 391) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| PV_In | Process value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_Unit | Unit of measure for process value | INT | 1001 |
| PV_OpScale* | OS display range for process value | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SP_DnRaLim* | Limit (low) for the gradient of the setpoint [Unit/Time unit]<br>Time unit depends on TimeFactor | REAL | 100.0 |
| SP_Ext | external setpoint - (to interconnection) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_ExtLi | 1 = Select internal setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtOp* | 1 = Select external setpoint (via operator) | BOOL | 0 |
| SP_HiLim | Limit (high) of setpoint | REAL | 100.0 |
| SP_LiDnRaLim | Linkable limit (low) for the gradient of the setpoint [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_LiRmpTime | Linkable time [Time unit] for gradient limitation in ramp mode duration<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_LiUpRaLim | Linkable gradient limit (high) for the setpoint [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_LoLim | Limit (low) of setpoint | REAL | 0.0 |
| SP_Int* | Internal setpoint for operation | REAL | 0.0 |
| SP_IntLi | 1 = Select internal setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_IntOp* | 1 = Select internal setpoint (via operator) | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP_LiOp | Select internal/external setpoint source:<br>1 = Via interconnection<br>0 = Via operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_OpScale | OS display range for setpoint | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>100.0<br>0.0 |
| SP_RateOn | 1 = Activate limitation of setpoint gradients | BOOL | 0 |
| SP_RmpModTime | 1 = Use time values (SP_RmpTime, SP_LiRmpTime)<br>0 = Use gradient limits (SP_DnRaLim, SP_LiDnRaLim, SP_UpRaLim, SP_LiUpRaLim) | BOOL | 0 |
| SP_RmpOn* | 1 = Activate setpoint ramp to target setpoint SP_RmpTarget | BOOL | 0 |
| SP_RmpTarget* | Target setpoint for setpoint ramp | REAL | 0.0 |
| SP_RmpTime* | Time [Time unit] for setpoint ramp from current SP up to SP_RmpTarget or time [Time unit] for gradient limitation in ramp mode duration<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_TrkExt | 1 = Bumpless switchover from external to internal setpoint active | BOOL | 0 |
| SP_Unit | Unit of measure for setpoint | INT | 1001 |
| SP_UpRaLim* | Gradient limit (high) for the setpoint [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 100.0 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | - |
| StepNo | Batch step number | DWORD | 16#00000000 |
| TimeFactor | Time unit:<br>0 = Seconds<br>1 = Minutes<br>2 = Hours | INT | 0 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |
| StateTrkOn | Initiate tracking of state | BOOL | 0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see OpAnL error handling (Page 394) | INT | -1 |
| MsgAckn | Message acknowledgment status (output ACK_STATE of ALARM_8P) | WORD | 16#0000 |
| MsgErr | 1 = Message error (output ERROR of ALARM_8P) | BOOL | 0 |
| MsgStat | Message status (output STATUS of ALARM_8P) | WORD | 16#0000 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| SP_Out | Setpoint used by controller | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_DnRaAct | 1 = Negative gradient limiting of setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtAct | 1 = External setpoint is active<br>0 = Internal setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_HiAct | High limit for SP_Ext or SP_Int reached or exceeded | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_LoAct | Low limit for SP_Ext or SP_Int reached or undershot | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_OpScaleOut | SP_OpScale as output | STRUCT<br>• High REAL<br>• Low: BYTE | <br>• 100.0<br>• 0.0 |
| SP_RateTarget | Target setpoint for the gradient limitation | REAL | 0.0 |
| SP_UpRaAct | 1 = Positive gradient limiting of setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_UnitOut | SP_Unit as output | INT | 0 |
| SP_RemRT | Remaining ramp time of the setpoint | REAL | 0.0 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word (Page 389) | DWORD | 16#00000000 |

## See also

OpAnL messaging (Page 395)
OpAnL block diagram (Page 403)
OpAnL modes (Page 390)

## 3.2.7 OpAnL block diagram

### OpAnL block diagram

A block diagram is not provided for this block.

### See also

OpAnL I/Os (Page 397)

OpAnL error handling (Page 394)

OpAnL functions (Page 391)

Description of OpAnL (Page 389)

OpAnL modes (Page 390)

OpAnL messaging (Page 395)

## 3.2.8 Operator control and monitoring

### 3.2.8.1 OpAnL views

### Views of the OpAnL block

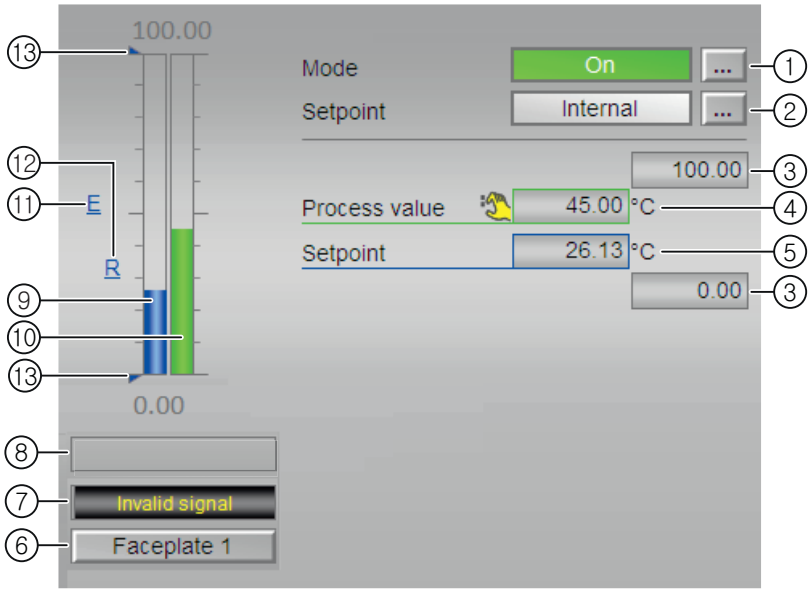
The block OpAnL provides the following views:

- OpAnL standard view (Page 404)
- Alarm view (Page 336)
- Trend view (Page 341)
- Ramp view (Page 334)
- OpAnL parameter view (Page 407)
- OpAnL preview (Page 409)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for OpAnL (Page 410)

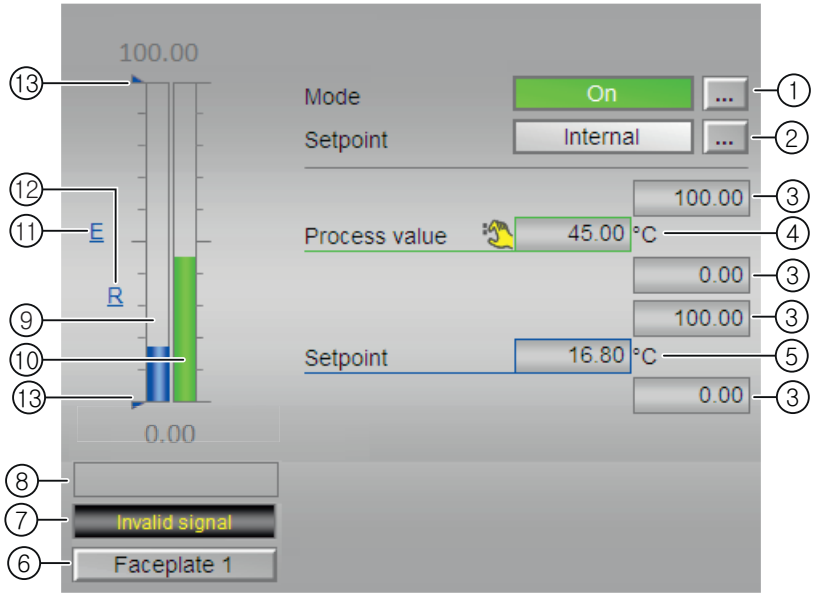
Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 3.2.8.2 OpAnL standard view

#### OpAnL standard view




The view is switched depending on the `Feature.Bit16` = 1 (process value with separate scale range)



### (1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Bar graph (Page 376)
- Out of service (Page 77)

Refer to chapter Switching operating states and operating modes (Page 289) for information on switching the operating mode.

### (2) Display and switch the setpoint specification

This area shows how to specify the setpoint. The setpoint can be specified as follows:

- By the application ("External", CFC/SFC)
- By the user directly in the faceplate ("Internal").

Refer to the Switching operating states and operating modes (Page 289) chapter for information on switching the setpoint specification.

You can find additional information on this in chapter Setpoint specification - internal/external (Page 136).

### (3) High and low scale range for the process value and setpoint

These values provide information on the display range for the bar graph of the process value. The scale range is defined in the Engineering System. For more information refer Bar graph (Page 376) section.

### (4) Display of the process value including signal status

This area shows the current process value with the corresponding signal status.

### (5) Display and change the setpoint including signal status

This area shows the current setpoint with the corresponding signal status.

Refer to the Changing values (Page 291) chapter for information on changing the setpoint. The setpoint specification also needs to be set to "Internal" for this block.

### (6) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the Engineering System (ES). The visibility of this navigation button depends on the configuration in the Engineering System (ES).

Refer also to chapter Opening additional faceplates (Page 234) for more on this.

### (7) Display area for block states

This area provides additional information on the operating state of the block:

- "Internal error"
- "Invalid Signal"

### (8) Display area for block states

This area provides additional information on the operating state of the block:

- "SP ramp active"

### (9) Bar graph for the process value

This area shows the current process value in the form of a bar graph. You can also see the high and low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (10) Bar graph for the setpoint

This area shows the current setpoint in the form of a bar graph. You can also see the high and low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (11) Display of external setpoint

This display [E] is only visible when you have selected "Internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

### (12) Display for the target setpoint of the setpoint ramp

This display [R] shows the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 334).

### (13) Displaying the limits

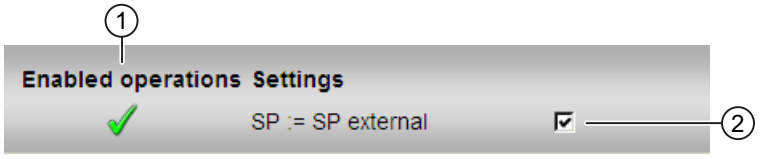
These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the Engineering System (ES).

### See also

On (Page 77)

### 3.2.8.3 OpAnL parameter view

#### Parameter view of OpAnL




#### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

#### (2) Settings

You can select the following functions in this area:

- "SP := SP external": ☑ Bumpless switchover of setpoint from external to internal. The internal setpoint is tracked to the external one.

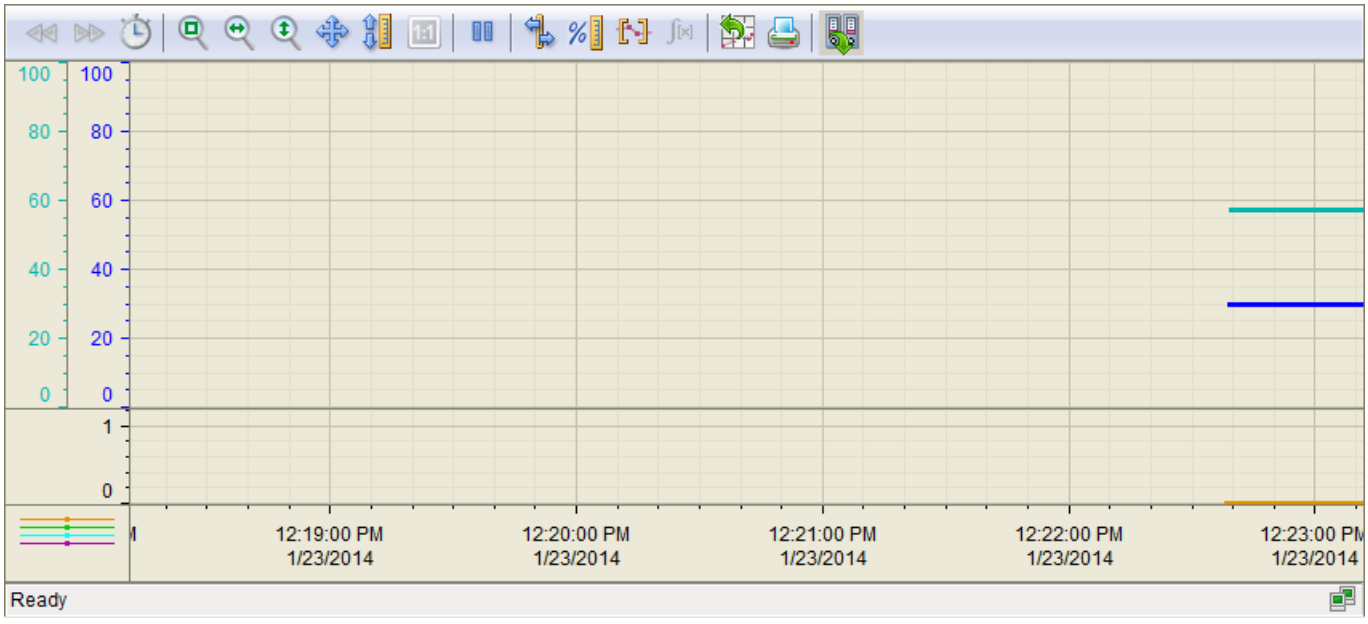
### 3.2.8.4 OpAnL trend view

#### OpAnL trend view

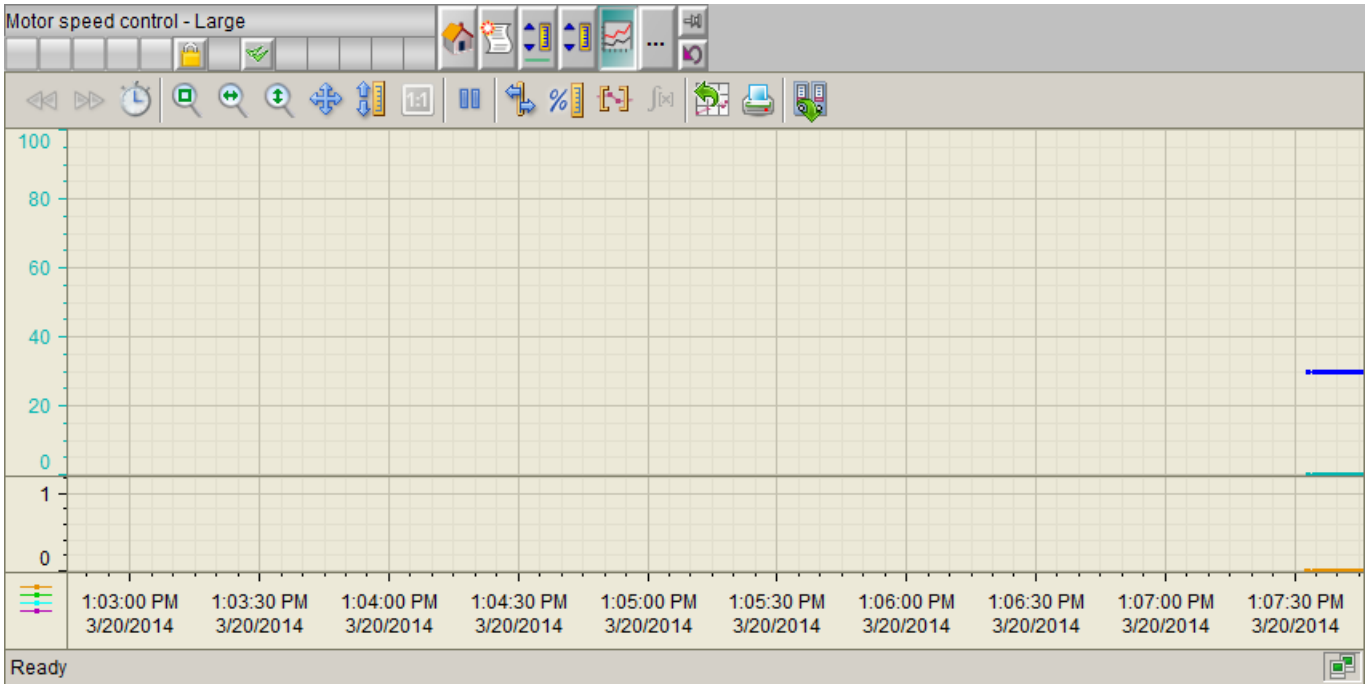
You can find general information in section Trend view (Page 341).

Depending on `Feature.Bit16` (process value with separate scale range) either one or two value axes are shown in the trend view.

- Process value with separate scale range




- Process value without separate scale range

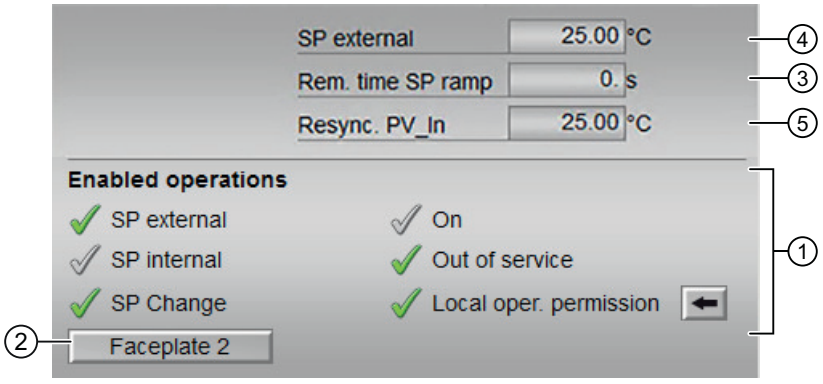


**Note**

This function only has an effect when `SP_Out` is configured in "TrendConfiguration1" and `PV_In` is configured in "TrendConfiguration2" in the block icon (default).

### 3.2.8.5 OpAnL preview

#### Preview of OpAnL




#### (1) Enable operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "SP external": You can feedforward the external setpoint.
- "SP internal": You can feedforward the internal setpoint.
- "Change SP": You can change the setpoint.
- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operator permission": Use the ← button to switch to the standard view of the OpStations block. You can find additional information on this in the section Operator control permissions (Page 285).

### (2) Navigation button for switching to the standard view of any faceplate

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

See also the section Opening additional faceplates (Page 234) for more on this.

### (3) Remaining time to reach SP ramp

- "Rem. time SP ramp" : Remaining time to reach the ramp target value.

### (4) SP external

Currently applicable external setpoint with the corresponding signal status.

### (5) Resynchronization

Displays the Resynchronization state based on readback value.

## 3.2.8.6 Block icon for OpAnL

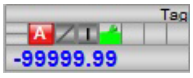
### Block icons for OpAnL

A variety of block icons are available with the following functions:

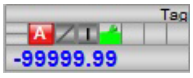
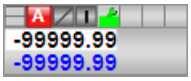
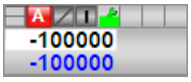
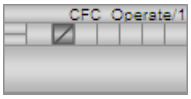
- Process tag type
- Operating modes
- Internal and external setpoint specification
- Signal status, release for maintenance
- Memo display
- Process value (black, with and without decimal places)
- Setpoint (blue, with and without decimal places)

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 3 | |
|  | 4 | |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | |
|  | 6 | |
|  | 7 | |
|  | 8 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

# 3.3 OpAnS - Check and output analog signals (small)

## 3.3.1 Description of OpAnS

### Object name (type + number) and family

Type + number: FB 1915

Family: Operate

### Area of application for OpAnS

The block is used for the following applications:

- Checking and transferring analog input values

### How it works

The block forwards incoming, internal (entered in the faceplate) or external (CFC/SFC) analog signals at the `SP_Int` or `SP_Ext` input to the `SP` output, depending on the setting for the `SP_LiOp` input parameter.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

### Status word allocation for `Status1` parameter

For a description of the individual parameters, see the chapter OpAnS I/Os (Page 417).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | Not used |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | Invalid signal status |
| 6 | `OnAct.Value` |
| 7 | Not used |
| 8 | `SP_ExtAct.Value` |

| Status bit | Parameter |
|---|---|
| 9 - 29 | Not used |
| 30 | Internal error |
| 31 | Not used |

### See also

OpAnS functions (Page 414)

OpAnS messaging (Page 417)

OpAnS block diagram (Page 420)

OpAnS error handling (Page 416)

OpAnS modes (Page 413)

## 3.3.2 OpAnS modes

### OpAnS operating modes

The block can be operated using the following modes:

- On (Page 77)
- Out of service (Page 77)

### "On"

General information on the "On" mode is available in the chapter On (Page 77).

### "Out of service"

You can find general information about the "Out of service" mode in the chapter Out of service (Page 77).

### See also

OpAnS I/Os (Page 417)

OpAnS messaging (Page 417)

OpAnS error handling (Page 416)

Description of OpAnS (Page 412)

OpAnS functions (Page 414)

OpAnS block diagram (Page 420)

## 3.3.3 OpAnS functions

### Functions of OpAnS

The functions for this block are listed below.

### Internal or external setpoint selection

This block provides the standard function Setpoint specification - internal/external (Page 136).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `SP_Out.ST`
- `PV_In.ST`
- `SP_Ext.ST` (only if `Feature.Bit10` = 1)

The signal status of the `SP_Out` output parameter is always equivalent to the signal status of input parameter `SP_Ext` or `SP_Int`, depending on how the setpoint is specified. If the internal setpoint `SP_Int` is used, the signal status is always output as 16#80.

In case of `Feature.Bit10` = 1, `SP_Ext.ST` influences `ST_Worst` independent of setpoint specification.

If an external setpoint is used, the signal status 16#60 (external simulation) is suppressed because the block acts as a sink with external simulation.

### Considering bad quality of automatic commands or external values

If the `Feature` bit Considering bad quality of automatic commands or external values (Page 200) is set to 1 and the parameter `SP_Ext` has bad signal status (16#00 or 16#28), the block works with the last valid value of `SP_Ext` in the "On" mode.

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 10 | Considering bad quality of automatic commands or external values (Page 200) |
| 24 | Enabling local operator authorization (Page 169) |

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 - 9 | Not used |
| 10 | 1 = Operator can switch to internal |
| 11 | 1 = Operator can switch to external |
| 12 | 1 = Operator can enable bumpless switchover |
| 13 | 1 = Operator can change `SP_Int` |
| 14 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Specifying the display area for process and setpoint values as well as operations

The block provides the standard function Display and operator input area for process values and setpoints (Page 233).

### Opening additional faceplates

The block provides the standard function Opening additional faceplates (Page 234).

### SIMATIC BATCH functionality

The block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of OpAnS (Page 412)
OpAnS messaging (Page 417)
OpAnS I/Os (Page 417)
OpAnS block diagram (Page 420)
OpAnS error handling (Page 416)
OpAnS modes (Page 413)
Using setpoint ramp (Page 131)
Gradient limit of the setpoint (Page 132)
Update acknowledgment and error status of the message call (Page 171)

## 3.3.4 OpAnS error handling

### Error handling of OpAnS

Refer to the chapter Error handling (Page 128) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 51 | Invalid signal<br>`SP_LiOp` = 1 and `SP_ExtLi` = 1 and `SP_IntLi` = 1 |

### See also

OpAnS block diagram (Page 420)
OpAnS I/Os (Page 417)
OpAnS messaging (Page 417)
OpAnS functions (Page 414)

OpAnS modes (Page 413)
Description of OpAnS (Page 412)

## 3.3.5 OpAnS messaging

### Messaging

This block does not offer messaging.

### See also

OpAnS error handling (Page 416)
OpAnS modes (Page 413)
OpAnS block diagram (Page 420)

## 3.3.6 OpAnS I/Os

### I/Os of OpAnS

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 414) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operating permission, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator control permissions (Page 414) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| PV_In | Process value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_Unit | Unit of measure for process value | INT | 1001 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SP_Ext | external setpoint - (to interconnection) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_ExtLi | 1 = Select internal setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtOp* | 1 = Select external setpoint (via operator) | BOOL | 0 |
| SP_Int* | Internal setpoint for operation | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP_IntLi | 1 = Select internal setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_IntOp* | 1 = Select internal setpoint (via operator) | BOOL | 1 |
| SP_LiOp | Select internal/external setpoint source:<br>1 = Via interconnection<br>0 = Via operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_OpScale | OS display range for setpoint | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>100.0<br>0.0 |
| SP_TrkExt | 1 = Bumpless switchover from external to internal setpoint active | BOOL | 0 |
| SP_Unit | Unit of measure for setpoint | INT | 1001 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see OpAnS error handling (Page 416) | INT | -1 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| SP_Out | Setpoint used by controller | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_OpScaleOut | SP_OpScale as output | STRUCT<br>• High REAL<br>• Low: BYTE | <br>• 100.0<br>• 0.0 |
| SP_UnitOut | SP_Unit as output | INT | 0 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word (Page 412) | DWORD | 16#00000000 |

### See also

OpAnS messaging (Page 417)

OpAnS block diagram (Page 420)

OpAnS modes (Page 413)

## 3.3.7 OpAnS block diagram

### OpAnS block diagram

A block diagram is not provided for this block.

### See also

OpAnS I/Os (Page 417)

OpAnS error handling (Page 416)

OpAnS functions (Page 414)

Description of OpAnS (Page 412)

OpAnS modes (Page 413)

OpAnS messaging (Page 417)

## 3.3.8 Operator control and monitoring

### 3.3.8.1 OpAnS views

#### Views of the OpAnS block

The block OpAnS provides the following views:

- OpAnS standard view (Page 421)
- Trend view (Page 341)
- OpAnS parameter view (Page 423)
- OpAnS preview (Page 424)
- Memo view (Page 338)
- Batch view (Page 336)
- OpAnS block icon (Page 425)

Refer to the chapters Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

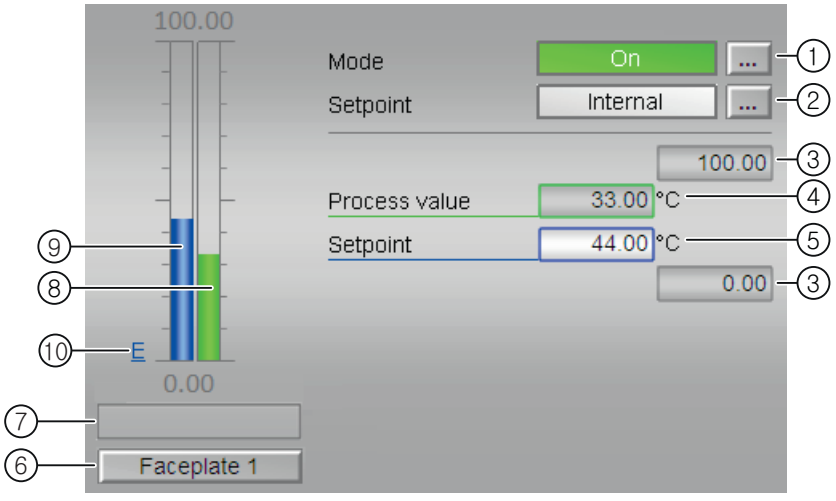
#### See also

Ramp view (Page 334)

Alarm view (Page 336)

### 3.3.8.2 OpAnS standard view

#### OpAnS standard view



### (1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 77)
- Out of service (Page 77)

Refer to chapter Switching operating states and operating modes (Page 289) for information on switching the operating mode.

### (2) Display and switch the setpoint specification

This area shows how to specify the setpoint. The setpoint can be specified as follows:

- By the application ("External", CFC/SFC)
- By the user directly in the faceplate ("Internal").

Refer to the Switching operating states and operating modes (Page 289) chapter for information on switching the setpoint specification.

You can find additional information on this in chapter Setpoint specification - internal/external (Page 136).

### (3) High and low scale range for the process value

These values provide information on the display range for the bar graph of the process value. The scale range is defined in the Engineering System.

### (4) Display of the process value including signal status

This area shows the current process value with the corresponding signal status.

### (5) Display and change the setpoint including signal status

This area shows the current setpoint with the corresponding signal status.

Refer to the Changing values (Page 291) chapter for information on changing the setpoint. The setpoint specification also needs to be set to "Internal" for this block.

### (6) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the Engineering System (ES). The visibility of this navigation button depends on the configuration in the Engineering System (ES).

Refer also to chapter Opening additional faceplates (Page 234) for more on this.

### (7) Display area for block states

This area provides additional information on the operating state of the block:

- "Internal error"
- "Invalid Signal"

### (8) Bar graph for the process value

This area shows the current process value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (9) Bar graph for the setpoint

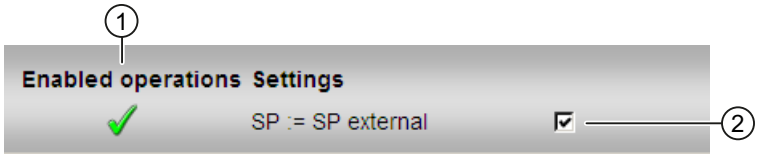
This area shows the current setpoint in the form of a bar graph. The visible area in the bar graph depends on the configuration in the Engineering System (ES).

### (10) Display of external setpoint

This display [E] is only visible when you have selected "Internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

## 3.3.8.3 OpAnS parameter view

### Parameter view of OpAnS




### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

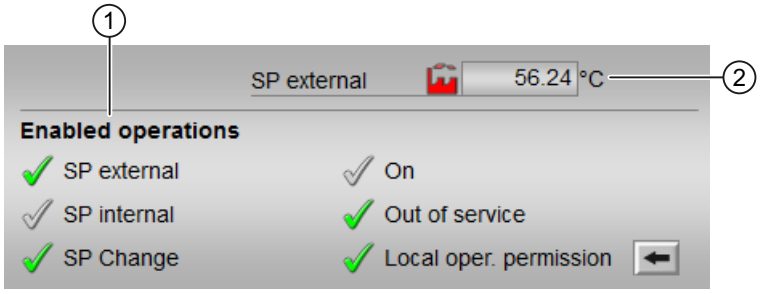
### (2) Settings

You can select the following functions in this area:

- "SP := SP external": ☑ Bumpless switchover of setpoint from external to internal. The internal setpoint is tracked to the external one.

## 3.3.8.4 OpAnS preview

### Preview of OpAnS




### (1) Enabled operations

This area shows all operations for which special operating permissions are assigned. They depend on the configuration in the Engineering System (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter at all due to the configured AS operating permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "SP external": You can feedforward the external setpoint.
- "SP internal": You can feedforward the internal setpoint.
- "Change SP": You can change the setpoint.
- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. You can find additional information on this in chapter Operator control permissions (Page 285).

### (2) SP external

Currently applicable external setpoint with the corresponding signal status.

### See also

Opening additional faceplates (Page 234)

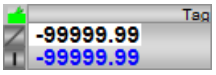
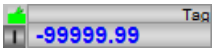
## 3.3.8.5 OpAnS block icon

### Block icons for OpAnS

A variety of block icons are available with the following functions:

- Process tag type
- Operating modes
- Internal and external setpoint specification
- Signal status, release for maintenance
- Memo display
- Process value (black, with and without decimal places)
- Setpoint (blue, with and without decimal places)

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | Block icon in the full display |
|  | 6 | Block icon in the full display |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

# 3.4 OpDi01 - Manipulating a digital value (2 pushbuttons)

## 3.4.1 Description of OpDi01

### Object name (type + number) and family

Type + number: FB 1866

Family: Operate

### Area of application for OpDi01

The block is used for the following applications:

- Manipulating a digital value

### How it works

A digital value is manipulated by interconnection or via the faceplate.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

If `Feature.Bit0` = 0, the output parameter `Out` will reset to 0.

### Status word allocation for `Status` parameter

You can find a description for each parameter in section OpDi01 I/Os (Page 432).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | Not used |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | Not used |
| 6 | `OnAct.Value` |
| 7 | `Out.Value` |
| 8 | `LiOp` |
| 9 | `FbkIn.Value` |
| 10 | Resynchronization preview for Out, Out1 and SP_Out |
| 11 | Resynchronization: Display preview |

| Status bit | Parameter |
|---|---|
| 12 - 13 | Not used |
| 14 | 1 = Invalid signal status |
| 15 | Not used |
| 16 | 0: Open padlock in the block icon<br>1: Closed padlock in the block icon |
| 17 | Hidden bypass signal in interlock |
| 18 | `Feature2.Bit2`: Separate bypass signal |
| 19 - 22 | Not used |
| 23 | "Interlock" button is enabled |
| 24 - 25 | Not used |
| 26 | Bypass information for display in block icon |
| 27 - 29 | Not used |
| 30 | Internal error |
| 31 | Not used |

**See also**

OpDi01 functions (Page 428)

OpDi01 messaging (Page 431)

OpDi01 block diagram (Page 435)

OpDi01 error handling (Page 431)

OpDi01 modes (Page 427)

## 3.4.2 OpDi01 modes

### OpDi01 operating modes

The block can be operated using the following modes:

- On (Page 77)
- Out of service (Page 77)

### "On"

General information on the "On" mode is available in the section On (Page 77).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

**See also**

OpDi01 block diagram (Page 435)
OpDi01 I/Os (Page 432)
OpDi01 messaging (Page 431)
OpDi01 error handling (Page 431)
OpDi01 functions (Page 428)
Description of OpDi01 (Page 426)

## 3.4.3 OpDi01 functions

### Functions of OpDi01

The functions for this block are listed below.

### Internal or external digital value

Use the `LiOp` input parameter to define whether the digital value is set (0 - 1, parameter `SetOp` or `SetLi`) or is reset (1 - 0, parameter `RstOp` or `RstLi`) via the faceplate or an interconnection.

`LiOp` = 0: Specification of digital value via faceplate (`SetOp` or `RstOp`)

`LiOp` = 1: Specification of digital value via interconnection (`SetLi` or `RstLi`)

### Interlocks

Use the `Intl_En` = 1 and `Intlock.ST` ≠ 16#FF input parameters to activate the interlock function on this block.

An active interlock condition brings the block to the neutral position (`Intlock.Value` = 0 or `Intlock.ST` = 16#00 input). Output parameter `Out` is set to 0. When the interlocking condition no longer applies, the digital value currently valid is output again, depending on the input parameters `LiOp`, `SetOp`, `RstOp`, `SetLi` and `RstLi`.

### Input parameter for feedback value

This block has a `FbkIn` input parameter for displaying a feedback value in the faceplate.

### Opening additional faceplates

This block provides the standard function Resynchronization (Page 258).

### Changing labels on buttons and text

This block provides the standard function Labeling of buttons and text (Page 235).

Instance-specific text can be configured for the following parameters:

- Out
- SetOp
- RstOp
- FbkIn

### Forming the signal status for blocks

The worst signal status ST_Worst for the block is formed from the following parameters:

- FbkIn.ST
- SetLi.ST (only if Feature.Bit10 = 1)
- RstLi.ST (only if Feature.Bit10 = 1)

### Considering bad quality of automatic commands or external values

If the Feature bit Considering bad quality of automatic commands or external values (Page 200) is set to 1 and one of the following parameters has bad signal status (16#00 or 16#28), the block goes to the neutral position (Out = 0) in the "On" mode:

- SetLi.ST
- RstLi.ST

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the OS_Perm parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can set the digital value |
| 5 | 1 = Operator can reset the digital value |
| 6 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in OS_Perm as a parameter, you have to reset the corresponding OS_Perm bit.

### Resynchronization

This block provides the standard function Resynchronization (Page 258).

Resynchronization is switched on over feature bits:

| | |
|---|---|
| Feature Bit17 =1 | Resynchronization after CPU startup |
| Feature Bit18 =1 | Resynchronization after Out of service |
| Feature Bit20 =1 | Resynchronization with 0->1 trigger at StateTrkOn input |

If there are no restrictions the control outputs are tracked as follow:

Out := FbkIn;

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the section Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) |
| 10 | Considering bad quality of automatic commands or external values (Page 200) |
| 17 | Resynchronization after startup (Page 213) |
| 18 | Resynchronization after Out of service (Page 213) |
| 19 | Resynchronization after CSF (Page 214) |
| 20 | Resynchronization with StateTrkOn input (Page 214) |
| 24 | Enabling local operator authorization (Page 169) |

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of OpDi01 (Page 426)

OpDi01 messaging (Page 431)

OpDi01 I/Os (Page 432)

OpDi01 block diagram (Page 435)

OpDi01 error handling (Page 431)

OpDi01 modes (Page 427)

Opening additional faceplates (Page 234)

## 3.4.4 OpDi01 error handling

### Error handling of OpDi01

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; block is not processed |
| 0 | There is no error. |
| 51 | `SetLi` = 1 and `RstLi` = 1 and `LiOp` = 1 |

### See also

OpDi01 block diagram (Page 435)

OpDi01 I/Os (Page 432)

OpDi01 messaging (Page 431)

OpDi01 functions (Page 428)

OpDi01 modes (Page 427)

Description of OpDi01 (Page 426)

## 3.4.5 OpDi01 messaging

### Messaging

This block does not offer messaging.

### See also

Description of OpDi01 (Page 426)

OpDi01 functions (Page 428)

OpDi01 I/Os (Page 432)

OpDi01 block diagram (Page 435)

OpDi01 error handling (Page 431)
OpDi01 modes (Page 427)

## 3.4.6 OpDi01 I/Os

### I/Os of OpDi01

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkTypeData* | This input can not be edit or linked and is for internal usage only! | | |
| EN | 1 = Called block will be processed | BOOL | 1 |
| FbkIn | Input for feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Feature | I/O for additional functions (Page 428) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 0<br>• 0 |
| Intlock | 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared<br>1 = Interlock not activated | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Intl_En | 1 = Interlock without reset (interlock, parameter Intlock) can be used | BOOL | 1 |
| LiOp | Switchover of operating mode between:<br>1 = Interconnection<br>0 = Operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosLi | Edge transition (0-1) = "Out of service", via interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operating permission, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator control permissions (Page 428) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| RstLi | Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | Reset by the operator | BOOL | 0 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | - |
| SetLi | Connected digital input | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SetOp* | Digital input for operator | BOOL | 0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| StateTrkOn | Initiate tracking of state | BOOL | 0 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see OpDi01 error handling (Page 431) | INT | -1 |
| IntlockOut | Intlock as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LockAct | 1 = Interlock (Intlock) is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| Out | Digital output value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status | Status word (Page 426) | DWORD | 16#00000000 |

## See also

OpDi01 messaging (Page 431)

OpDi01 block diagram (Page 435)

OpDi01 modes (Page 427)

## 3.4.7 OpDi01 block diagram

### OpDi01 block diagram

A block diagram is not provided for this block.

### See also

OpDi01 I/Os (Page 432)

OpDi01 messaging (Page 431)

OpDi01 error handling (Page 431)

OpDi01 functions (Page 428)

OpDi01 modes (Page 427)

Description of OpDi01 (Page 426)

## 3.4.8 Operator control and monitoring

### 3.4.8.1 OpDi01 views

### Views of the OpDi01 block

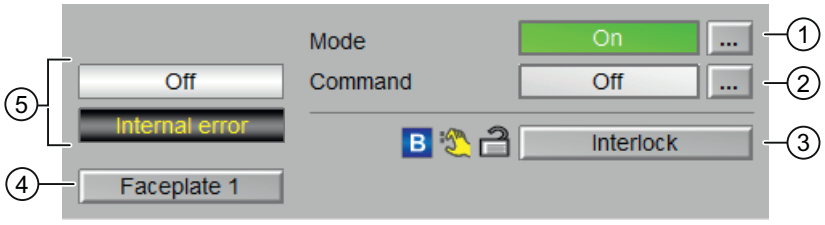
The block OpDi01 provides the following views:

- OpDi01 standard view (Page 436)
- Trend view (Page 341)
- OpDi01 preview (Page 437)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for OpDi01 (Page 439)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 3.4.8.2 OpDi01 standard view

#### OpDi01 standard view




#### (1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 77)
- Out of service (Page 77)

You can find additional information on this in section Switching operating states and operating modes (Page 289).

#### (2) Displaying and switching the command

This area shows you the current selection. You can output a continuous signal as follows:

- "On": Continuous signal is output
- "Off"

You can find additional information on this in section Switching operating states and operating modes (Page 289).

You can rename the display text as you please, as described in the section Labeling of buttons and text (Page 235).

Do this with the following parameters:

- Text for "Command": Parameter `SetOp#string_1`
- Text for "On/Off": Parameter `Out#string_0` / `Out#string_1`

#### (3) Operating range for the interlock functions of the block

This display is only visible when the corresponding block input is interconnected.

You can use this button to control the interlock functions of the block. You can find additional information on this in section OpDi01 functions (Page 428).

- Bypass information (see Forming the group status for interlock information (Page 113)):



If the text is configured for the interlocks, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (4) Button for switching to the standard view of any faceplate

Use this button for the standard view of a block configured in the engineering system. The visibility of this button depends on the configuration in the engineering system (ES).

You can find additional information on this in section Opening additional faceplates (Page 234).

### (5) Displaying the feedback of the command

This area shows you the currently valid command. The following commands can be shown here:

- "On"
- "Off"
- "Internal error"
- "Invalid Signal"

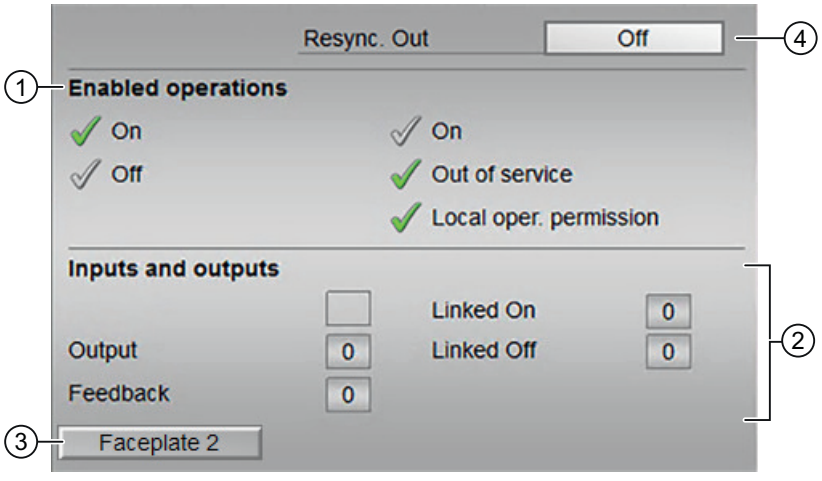
You can rename the display text as you please, as described in the section Labeling of buttons and text (Page 235).

Do this with the following parameter:

- Text for "On/Off": Parameter `FbkIn#string_0` / `FbkIn#string_1`

## 3.4.8.3 OpDi01 preview

### Preview of OpDi01




### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "On": You can set the digital value (0 - 1 edge).
  You can rename the display text as you please, as described in the section Labeling of buttons and text (Page 235).
  Do this with the following parameters:
  - Text for "On": Parameter `.Out#Value#string_1`
- "Off": You can set the digital value (1 - 0 edge).
  You can rename the display text as you please, as described in the section Labeling of buttons and text (Page 235).
  Do this with the following parameters:
  - Text for "Off": Parameter `.Out#Value#string_0`
- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. Additional information is available in the section Operator control permissions (Page 285).

### (2) Display of current inputs and outputs

This area shows the most important parameters for this block with the current selection:

- "Interlock":
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Output": 1 = Digital output value set
- "Feedback": 1 = Feedback set
- "Linked On": 1 = Linked input on
- "Linked Off": 1 = Linked input off

### (3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the Engineering System (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

Refer also to the Opening additional faceplates (Page 234) section for more on this.

### (4) Resynchronization

Displays the Resynchronization state based on feedback value.

## 3.4.8.4 Block icon for OpDi01

### Block icons for OpDi01

A variety of block icons are available with the following functions:

- Process tag type
- Operating modes
- Signal status, release for maintenance
- Bypass
- Interlocks
- Output signal
- Memo display

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|   | 1 | |
|   | 2 | |
|   | 3 | Block icon in the full display |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|   | 1 | Block icon in the full display |
|   | 2 | |

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 3 | |
|  | 4 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269).

# 3.5 OpDi03 - Manipulating a digital value (3 pushbuttons)

## 3.5.1 Description of OpDi03

### Object name (type + number) and family

Type + number: FB 1867

Family: Operate

### Area of application for OpDi03

The block is used for the following applications:

- Manipulating a digital value (3 pushbuttons)

### How it works

A digital value is manipulated at three possible outputs by interconnection or via the faceplate.

If two or three input parameters are set for an interconnection (parameter `SetLix`), the input parameter with the highest index will be set to the corresponding output parameter. For example, if the `SetLi1` and `SetLi2` input parameters are set (= 1), then `Out2` will be set (= 1).

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

If `Feature.Bit0` = 0, the output parameters `Out1` to `Out3` will reset to 0.

### Status word allocation for `Status` parameter

You can find a description for each parameter in section OpDi03 I/Os (Page 448).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | Not used |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | Not used |
| 6 | `OnAct.Value` |
| 7 | `Out1.Value` |
| 8 | `Out2.Value` |

| Status bit | Parameter |
|---|---|
| 9 | `Out3.Value` |
| 10 | `LiOp.Value` |
| 11 | `Fbk1In.Value` |
| 12 | `Fbk2In.Value` |
| 13 | `Fbk3In.Value` |
| 14 | 1 = Invalid signal status |
| 15 | Not used |
| 16 | 0: Open padlock in the block icon<br>1: Closed padlock in the block icon |
| 17 | Hidden bypass signal in interlock |
| 18 | `Feature2.Bit2`: Separate bypass signal |
| 19 | Resynchronization preview for Out1 |
| 20 | Resynchronization preview for Out2 |
| 21 | Resynchronization preview for Out3 |
| 22 | Resynchronization: Display preview |
| 23 | "Interlock" button is enabled |
| 24 - 25 | Not used |
| 26 | Bypass information for display in block icon |
| 27 - 29 | Not used |
| 30 | Internal error |
| 31 | Not used |

### See also

OpDi03 functions (Page 443)
OpDi03 messaging (Page 447)
OpDi03 block diagram (Page 451)
OpDi03 error handling (Page 446)
OpDi03 modes (Page 442)

## 3.5.2 OpDi03 modes

### OpDi03 operating modes

The block can be operated using the following modes

- On (Page 77)
- Out of service (Page 77)

### "On"

You can find general information about the "On" mode in the On (Page 77) section.

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

### See also

OpDi03 block diagram (Page 451)

OpDi03 I/Os (Page 448)

OpDi03 messaging (Page 447)

OpDi03 error handling (Page 446)

OpDi03 functions (Page 443)

Description of OpDi03 (Page 441)

## 3.5.3 OpDi03 functions

### Functions of OpDi03

The functions for this block are listed below.

### Internal or external digital value

Use the `LiOp` input parameter to define whether it is through the faceplate or an interconnection using a 1 out of 3 that the selection of the digital value is set (0 - 1) or reset (1 - 0, input parameter `RstOut`).

- `LiOp` = 0: Specification of digital value via faceplate. One of the input parameters `SetOp1`, `SetOp2` or `SetOp3` is now routed to the relevant output `Out1`, `Out2` or `Out3`. For example, if `SetOp2` = 1 , then `Out2` = 1.
- `LiOp` = 1: Specification of digital value via interconnection. One of the input parameters `SetLi1`, `SetLi2` or `SetLi3` is now routed to the relevant output `Out1`, `Out2` or `Out3`. For example, if `SetLi2` = 1 , then `Out2` = 1.

The reset (1 - 0) is always undertaken using the `RstOut` input parameter.

### Interlocks

Use the `Intl_En` = 1 and `Intlock.ST` ≠ 16#FF input parameters to activate the interlock function on this block.

An active interlock condition brings the block to the neutral position (`Intlock.Value` = 0 or `Intlock.ST` = 16#00 input). Output parameter `Out` is set to 0. When the interlocking

condition no longer applies, the digital value currently valid is output again, depending on the input parameters `LiOp`, `SetOp`, `RstOp`, `SetLi` and `RstLi`.

### Input parameter for feedback value

This block has three input parameters `Fbk1In`, `Fbk2In` and `Fbk3In` for displaying three feedback values in the faceplate.

### Resetting all output values

Reset all output parameters (`Out1` ... `Out3`) by setting all interconnected input parameters for setting (`SetLi1` ... `SetLi3`) or enabled input parameters for setting (`SetOp1` ... `SetOp3`) to 0.

A 0 - 1 edge at the `RstOut` parameter then resets the three output parameters `Out1` ... `Out3`.

### Opening additional faceplates

This block provides the standard function Resynchronization (Page 258).

### Changing labels on buttons and text

This block provides the standard function Labeling of buttons and text (Page 235).

Instance-specific text can be configured for the following parameters:

- `OutX`
- `SetOpX`
- `FbkXIn`

X = (1 ... 3)

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `Fbk1In.ST`
- `Fbk2In.ST`
- `Fbk3In.ST`
- `SetLi1.ST` (only if `Feature.Bit10` = 1)
- `SetLi2.ST` (only if `Feature.Bit10` = 1)
- `SetLi3.ST` (only if `Feature.Bit10` = 1)
- `RstOut.ST` (only if `Feature.Bit10` = 1)

### Considering bad quality of automatic commands or external values

If the `Feature` bit Considering bad quality of automatic commands or external values (Page 200) is set to 1 and one of the following parameters has bad signal status (16#00 or 16#28), the block goes to the neutral position (`Out1` = 0, `Out2` = 0, and `Out3` = 0) in the "On" mode:

- `SetLi1.ST`
- `SetLi2.ST`
- `SetLi3.ST`
- `RstOut.ST`

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can set the digital value `SetOp1` |
| 5 | 1 = Operator can set the digital value `SetOp2` |
| 6 | 1 = Operator can set the digital value `SetOp3` |
| 7 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Resynchronization

This block provides the standard function Resynchronization.

Resynchronization is switched on over feature bits:

| | |
|---|---|
| Feature Bit17 =1 | Resynchronization after CPU startup |
| Feature Bit18 =1 | Resynchronization after Out of service |
| Feature Bit20 =1 | Resynchronization with 0->1 trigger at StateTrkOn input |

If there are no restrictions the control outputs are tracked as follow:

Out1 := Fbk1In;

Out2 := Fbk2In;

Out3 := Fbk3In;

If more than one feedback signals are active: The feedback with the highest index becomes active.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the section Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) |
| 10 | Considering bad quality of automatic commands or external values (Page 200) |
| 17 | Resynchronization after startup (Page 213) |
| 18 | Resynchronization after Out of service (Page 213) |
| 19 | Resynchronization after CSF (Page 214) |
| 20 | Resynchronization with StateTrkOn input (Page 214) |
| 24 | Enabling local operator authorization (Page 169) |

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of OpDi03 (Page 441)

OpDi03 messaging (Page 447)

OpDi03 I/Os (Page 448)

OpDi03 block diagram (Page 451)

OpDi03 error handling (Page 446)

OpDi03 modes (Page 442)

Opening additional faceplates (Page 234)

## 3.5.4 OpDi03 error handling

### Error handling of OpDi03

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; block is not processed |
| 0 | There is no error. |
| 51 | The value 1 is set at two or more inputs<br>(`SetLi1` = 1 AND `SetLi2` = 1) or (`SetLi2` = 1 AND `SetLi3` = 1) or (`SetLi1` = 1 AND SetLi3 = 1) and `LiOp` = 1 |

### See also

OpDi03 block diagram (Page 451)

OpDi03 I/Os (Page 448)

OpDi03 messaging (Page 447)

OpDi03 functions (Page 443)

OpDi03 modes (Page 442)

Description of OpDi03 (Page 441)

## 3.5.5 OpDi03 messaging

### Messaging

This block does not offer messaging.

### See also

Description of OpDi03 (Page 441)

OpDi03 functions (Page 443)

OpDi03 I/Os (Page 448)

OpDi03 block diagram (Page 451)

OpDi03 error handling (Page 446)

OpDi03 modes (Page 442)

## 3.5.6 OpDi03 I/Os

### I/Os of OpDi03

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| Fbk1In | Feedback for input In1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Fbk2In | Feedback for input In2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Fbk3In | Feedback for input In3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Feature | I/O for additional functions (Page 443) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 0<br>• 0 |
| Intlock | 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared<br>1 = Interlock not activated | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Intl_En | 1 = Interlock without reset (interlock, parameter Intlock) can be used | BOOL | 1 |
| LiOp | Switchover of operating mode between:<br>1 = Interconnection<br>0 = Operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operating permission, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator control permissions (Page 443) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| RstOut | Reset by the operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | - |
| SetLi1 | Connected digital input 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SetLi2 | Connected digital input 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SetLi3 | Connected digital input 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SetOp1* | Digital input 1 for operator | BOOL | 0 |
| SetOp2* | Digital input 2 for operator | BOOL | 0 |
| SetOp3* | Digital input 3 for operator | BOOL | 0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| StateTrkOn | Initiate tracking of state | BOOL | 0 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see OpDi03 error handling (Page 446) | INT | -1 |
| LockAct | 1 = Interlock (Intlock) is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IntlockOut | Intlock as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| Out1 | Digital output value 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Out2 | Digital output value 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Out3 | Digital output value 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status | Status word (Page 441) | DWORD | 16#00000000 |

**See also**

OpDi03 messaging (Page 447)

OpDi03 block diagram (Page 451)

OpDi03 modes (Page 442)

## 3.5.7 OpDi03 block diagram

### OpDi03 block diagram

A block diagram is not provided for this block.

**See also**

OpDi03 I/Os (Page 448)

OpDi03 messaging (Page 447)

OpDi03 error handling (Page 446)

OpDi03 functions (Page 443)

OpDi03 modes (Page 442)

Description of OpDi03 (Page 441)

## 3.5.8 Operator control and monitoring

### 3.5.8.1 OpDi03 view

### Views of the OpDi03 block

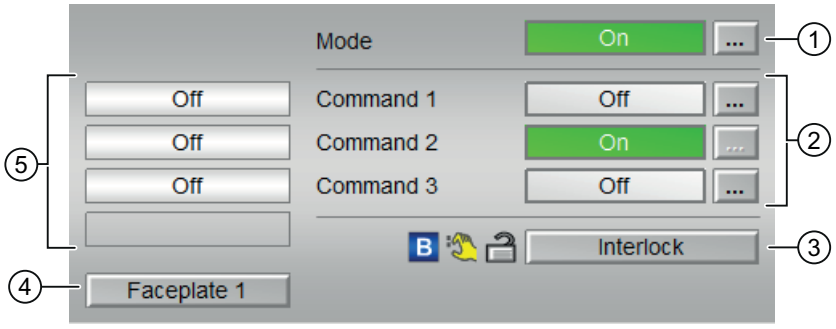
The block OpDi03 provides the following views:

- OpDi03 standard view (Page 452)
- Trend view (Page 341)
- OpDi03 preview (Page 454)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for OpDi03 (Page 455)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 3.5.8.2 OpDi03 standard view

#### OpDi03 standard view




#### (1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 77)
- Out of service (Page 77)

You can find additional information on this in section Switching operating states and operating modes (Page 289).

#### (2) Displaying and switching the command 1 to 3

This area shows you the current selection. You can output a continuous signal at the outputs `Out1` to `Out3` as follows:

- "On": Continuous signal is output
- "Off"

You can find additional information on this in section Switching operating states and operating modes (Page 289).

You can rename the display text as you please, as described in the section Labeling of buttons and text (Page 235).

**Note**

You can configure any of 3 commands available from the operation area of the faceplate.

Do this with the following parameters:

- Text for "Command 1/Command 2/Command 3": Parameter `SetOpX#string_1`, (X = 1 ... 3)
- Text for "On/Off": Parameter `OutX#string_0`, `OutX#string_1`, (X = 1 ... 3)

### (3) Operating range for the interlock functions of the block

This display is only visible when the corresponding block input is interconnected.

You can use this button to control the interlock functions of the block. You can find additional information on this in section OpDi03 functions (Page 443).

- Bypass information (see Forming the group status for interlock information (Page 113)):



If the text is configured for the interlocks, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (4) Button for switching to the standard view of any faceplate

Use this button for the standard view of a block configured in the engineering system. The visibility of this button depends on the configuration in the engineering system (ES).

You can find additional information on this in section Opening additional faceplates (Page 234).

### (5) Displaying the feedback of the command 1 to 3

This area shows you the current valid selection from `Out1` to `Out3`.

- "On"
- "Off"
- "Internal error"
- "Invalid Signal"

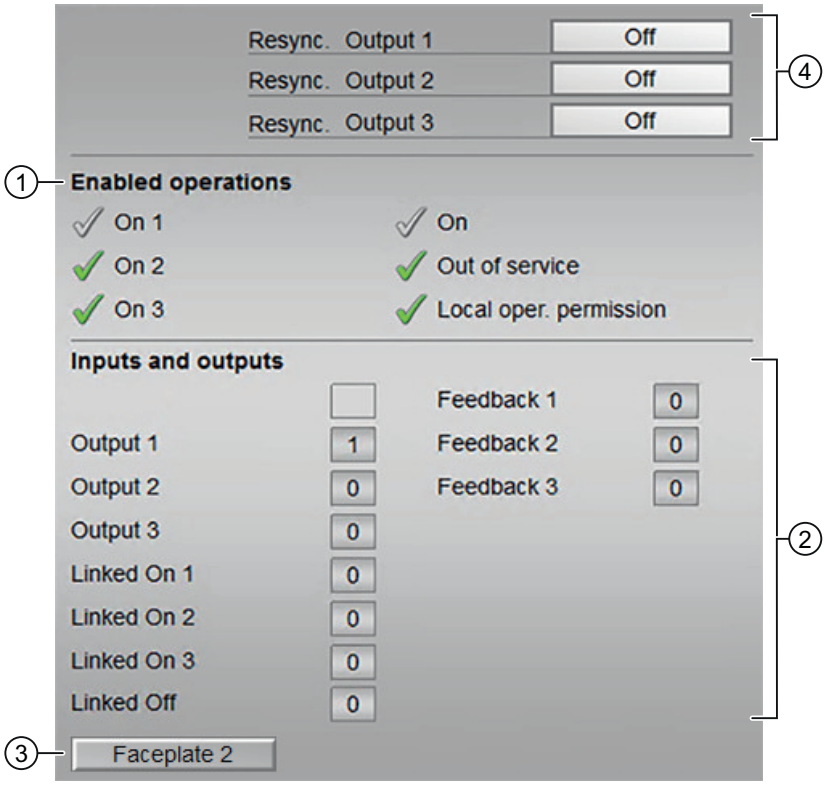
You can rename the display text as you please, as described in the section Labeling of buttons and text (Page 235).

Do this with the following parameters:

- Text for "On/Off": Parameter `FbkInX#string_1`, `FbkInX#string_0`, (X = 1 ... 3)

### 3.5.8.3 OpDi03 preview

**Preview of OpDi03**




**(1) Enabled operations**

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "On 1 to 3": You can now set the digital value (0 - 1 edge).
  You can rename the display text as you please, as described in the section Labeling of buttons and text (Page 235).
  Do this with the following parameters:
  - Text for "Command X": Parameter `OutX#string_1`, (X = 1 ... 3)
- "On": You can switch to "On" operating mode.

- "Out of service": You can switch to "Out of service" operating mode.
- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. Additional information is available in the section Operator control permissions (Page 285).

### (2) Display of current inputs and outputs

This area shows the most important parameters for this block with the current selection:

- "Interlock":
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Output 1 to 3": 1 = Digital output value set
- "Feedback 1 to 3": 1 = Feedback set
- "Linked On 1 to 3": 1 = Linked input on
- "Linked Off": 1 = Linked input off

### (3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

Refer also to the Opening additional faceplates (Page 234) section for more on this.

### (4) Resynchronization

Displays the Resynchronization state based on feedback value.

## 3.5.8.4 Block icon for OpDi03

### Block icons for OpDi03

A variety of block icons are available with the following functions:

- Process tag type
- Operating modes
- Signal status, release for maintenance
- Bypass
- Interlocks

- Output signal
- Memo display

The block icons from template @TemplateAPLV8.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | Block icon in the full display |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269).

# 3.6 OpStations - Configuration of the local operator authorization

## 3.6.1 Description of OpStations

### Object name (type + number) and family

Type + number: FB 1901

Family: Operate

### Area of application for OpStations

The block is used for the following applications:

- Configuration of the local operatorcontrol permission

### How it works

The block converts the enabled operations or locks for up to 16 individual permissions in the bit-coded `Out` output.

Refer also to OpStations block diagram (Page 464).

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The OpStations block and the technologic block must be installed into the same cyclic interrupt OB.

**Note**

Install the blocks in a slower cyclic interrupt OB.

With the default, the local operatorcontrol permission for all block instances on the faceplate is disabled (= 0) and is configured as follows:

1. Assign each operator station a bit-coded ID (1, 2, 4, 8, 16, ...), which you set in the internal APLOpStation variables as a start value. You can set up to 16 different operator stations. The variable is automatically created using the OS project editor and is located in the Split Screen Manager group.

   **Note**

   A redundant server pair can only be represented as an operator station, which also affects referenced client or single stations in the engineering station (ES). If a distinguishable operator station is needed, different groups of referenced clients or single stations must be created

2. Place the OpStations block in the chart.

   **Note**

   If you use the Opening additional faceplates (Page 234) function in multiple technologic blocks, you need to use one OpStations block for every technologic block. Otherwise, the call of the technologic block from the OpStations block is only possible for a technologic block.

3. Enable the function at the technologic block via `Feature.Bit24`(Enabling local operator authorization (Page 169)).
4. Interconnect the `OpSt_In` input parameter of the technologic block to the `Out` output parameter of the OpStations block.
5. Select the operator stations on which the technologic block will usually be used. You can select several at once. The selection is sent bit-coded to the `Out` output of the OpStations block.
6. Specify a list for the texts for the standard view in the OpStations faceplate in the shared declarations under the name APLOpStations in the SIMATIC Manager. Refer also to the section OpStations standard view (Page 467) for more information.

Once these configuration steps are completed, the local operatorcontrol permission is enabled. The enable for a technologic block is made when the bit-by-bit comparison between the `OpSt_Out` parameter and the operator station ID "APLOpStation" does not equal 0.

**Note**

The local operatorcontrol permission is not visualized in the "Enabled operation" icons of the block I/Os.

## Startup characteristics

This block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section OpStations I/Os (Page 462).

| Status bit | Parameter |
|---|---|
| 0 | `In0` |
| 1 | `In1` |
| 2 | `In2` |
| 3 | `In3` |
| 4 | `In4` |
| 5 | `In5` |
| 6 | `In6` |
| 7 | `In7` |
| 8 | `In8` |
| 9 | `In9` |
| 10 | `In10` |
| 11 | `In11` |
| 12 | `In12` |
| 13 | `In13` |
| 14 | `In14` |
| 15 | `In15` |
| 16 - 29 | Not used |
| 30 | `Occupied` |
| 31 | `BatchEn` |

### See also

OpStations operating modes (Page 459)

OpStations functions (Page 460)

OpStations error handling (Page 461)

OpStations messaging (Page 461)

## 3.6.2 OpStations operating modes

### OpStations operating modes

This block does not have any modes.

### See also

Description of OpStations (Page 457)

OpStations functions (Page 460)

OpStations error handling (Page 461)

OpStations messaging (Page 461)
OpStations I/Os (Page 462)
OpStations block diagram (Page 464)

## 3.6.3 OpStations functions

### Functions of OpStations

The functions for this block are listed below.

### Operator control permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can enable `In0` |
| 1 | 1 = Operator can enable `In1` |
| 2 | 1 = Operator can enable `In2` |
| 3 | 1 = Operator can enable `In3` |
| 4 | 1 = Operator can enable `In4` |
| 5 | 1 = Operator can enable `In5` |
| 6 | 1 = Operator can enable `In6` |
| 7 | 1 = Operator can enable `In7` |
| 8 | 1 = Operator can enable `In8` |
| 9 | 1 = Operator can enable `In9` |
| 10 | 1 = Operator can enable `In10` |
| 11 | 1 = Operator can enable `In11` |
| 12 | 1 = Operator can enable `In12` |
| 13 | 1 = Operator can enable `In13` |
| 14 | 1 = Operator can enable `In14` |
| 15 | 1 = Operator can enable `In15` |
| 16 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of OpStations (Page 457)
OpStations operating modes (Page 459)
OpStations error handling (Page 461)
OpStations messaging (Page 461)
OpStations I/Os (Page 462)
OpStations block diagram (Page 464)

## 3.6.4 OpStations error handling

### OpStations error handling

This block does not have any error handling.

### See also

Description of OpStations (Page 457)
OpStations operating modes (Page 459)
OpStations functions (Page 460)
OpStations messaging (Page 461)
OpStations I/Os (Page 462)
OpStations block diagram (Page 464)

## 3.6.5 OpStations messaging

### OpStations messaging

This block does not offer messaging.

### See also

Description of OpStations (Page 457)
OpStations operating modes (Page 459)
OpStations functions (Page 460)

OpStations error handling (Page 461)
OpStations I/Os (Page 462)
OpStations block diagram (Page 464)

## 3.6.6 OpStations I/Os

### OpStations I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 460) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| In0 | 1 = Enable for operator station 1 | BOOL | 0 |
| In1 | 1 = Enable for operator station 2 | BOOL | 0 |
| In2 | 1 = Enable for operator station 3 | BOOL | 0 |
| In3 | 1 = Enable for operator station 4 | BOOL | 0 |
| In4 | 1 = Enable for operator station 5 | BOOL | 0 |
| In5 | 1 = Enable for operator station 6 | BOOL | 0 |
| In6 | 1 = Enable for operator station 7 | BOOL | 0 |
| In7 | 1 = Enable for operator station 8 | BOOL | 0 |
| In8 | 1 = Enable for operator station 9 | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| In9 | 1 = Enable for operator station 10 | BOOL | 0 |
| In10 | 1 = Enable for operator station 11 | BOOL | 0 |
| In11 | 1 = Enable for operator station 12 | BOOL | 0 |
| In12 | 1 = Enable for operator station 13 | BOOL | 0 |
| In13 | 1 = Enable for operator station 14 | BOOL | 0 |
| In14 | 1 = Enable for operator station 15 | BOOL | 0 |
| In15 | 1 = Enable for operator station 16 | BOOL | 0 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OS_Perm | I/O for operator control permissions (Page 460) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| Out | Output value: for additional interconnections to the OpSt_In input of a technologic block | DWORD | 16#00000000 |
| Status | Status word (Page 457) | DWORD | 16#00000000 |

## See also

OpStations operating modes (Page 459)

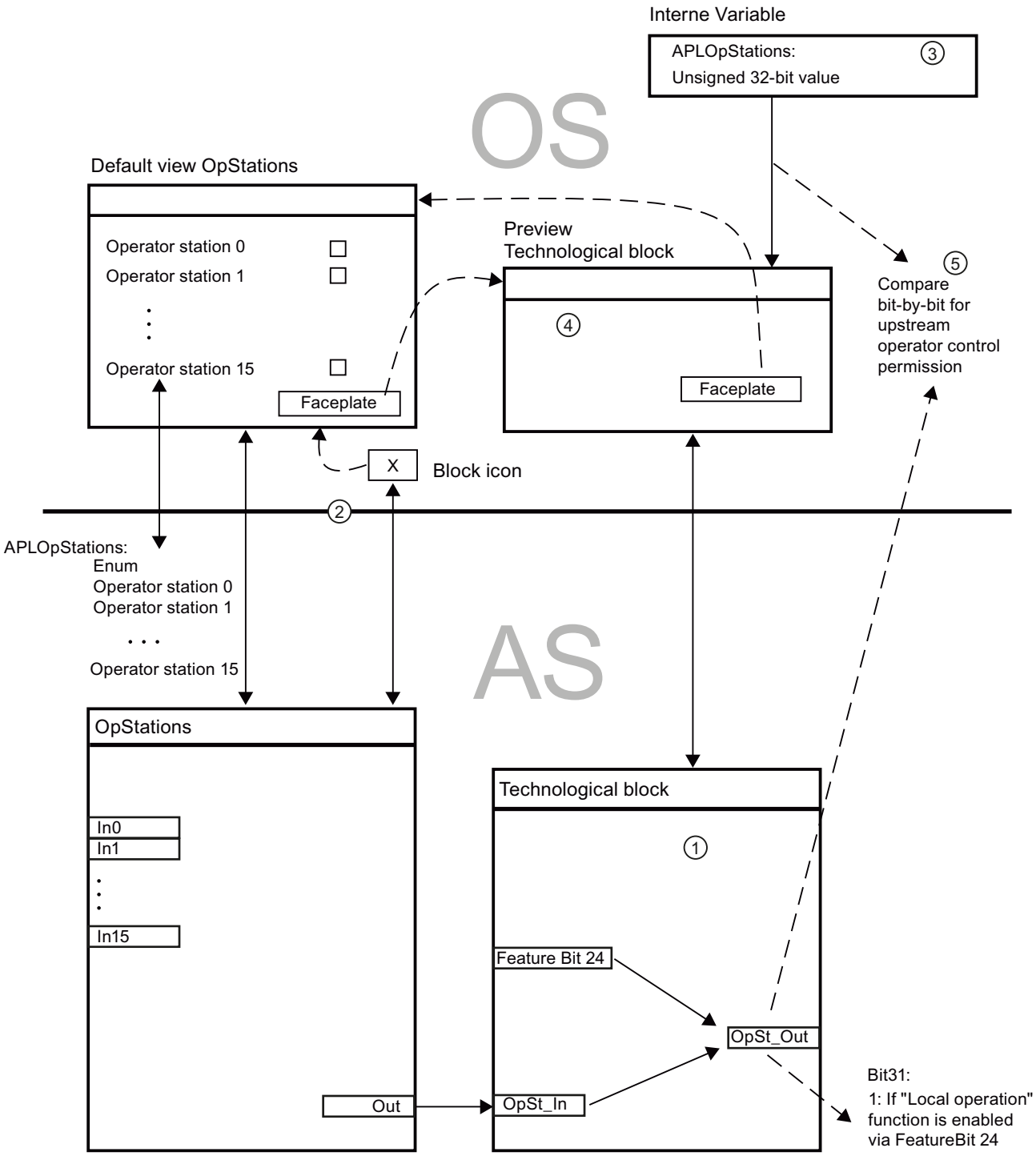
OpStations error handling (Page 461)

OpStations messaging (Page 461)

OpStations block diagram (Page 464)

## 3.6.7 OpStations block diagram

### OpStations block diagram


Interne Variable
APLOpStations:
Unsigned 32-bit value
3
OS
Default view OpStations
Operator station 0
Operator station 1
Operator station 15
Faceplate
Preview
Technological block
4
Faceplate
5
Compare
bit-by-bit for
upstream
operator control
permission
X
Block icon
2
APLOpStations:
Enum
Operator station 0
Operator station 1
Operator station 15
AS
OpStations
In0
In1
In15
Out
Technological block
1
Feature Bit 24
OpSt_Out
OpSt_In
Bit31:
1: If "Local operation"
function is enabled
via FeatureBit 24

**See also**

Description of OpStations (Page 457)
OpStations operating modes (Page 459)
OpStations functions (Page 460)
OpStations error handling (Page 461)
OpStations messaging (Page 461)
OpStations I/Os (Page 462)

## 3.6.8 Operator control and monitoring

### 3.6.8.1 OpStations views

**Views of the OpStations block**

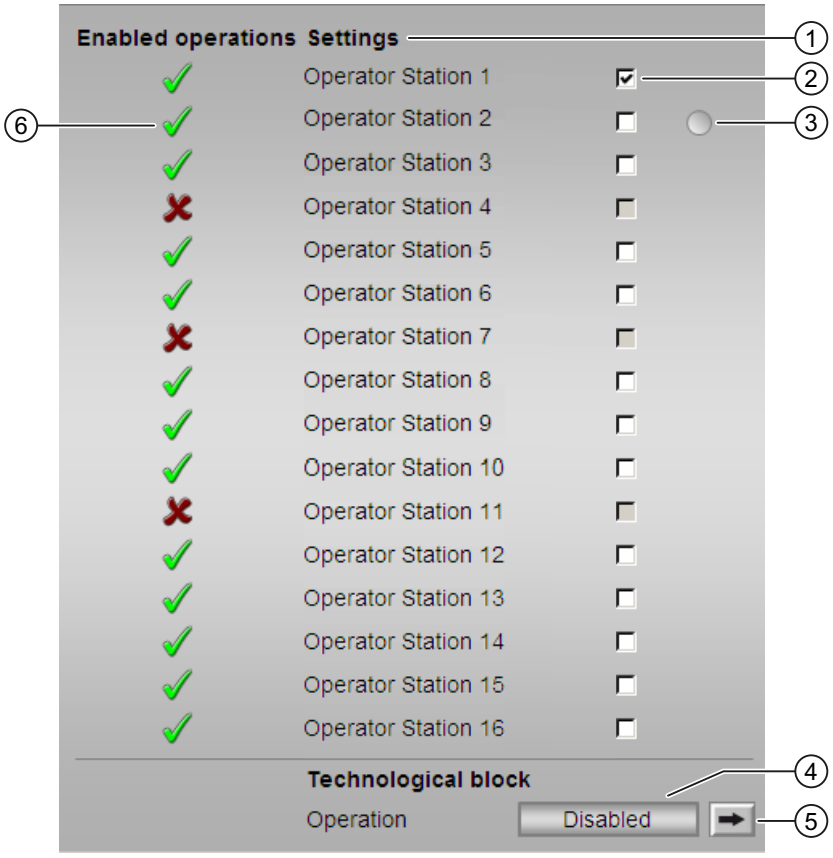
The `OpStations` block provides the following views:

- OpStations standard view (Page 467)
- Memo view (Page 338)
- Block icon of OpStations (Page 469)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 3.6.8.2 OpStations standard view

## OpStations standard view




## (1) Configurable display text under the settings

You yourself can define the text for operator station 1 to operator station 15 instead of the default text. Follow the steps outlined below:

- Create a list with the name "APLOpStations" in the SIMATIC Manager in the shared declarations. To learn more on this, refer to "How to save shared declarations" in the Process Control System PCS 7 Engineering System Configuration Manual.

Only values from 0 to 15 are permitted, other values will not be saved in the list.

The display names of the values are changed with the names of the operator stations. The display name corresponds to the value 0 of the display for the check box `In0` in the standard view etc.

If the text is not configured, the entire line is not displayed.

### (2) Disabling or enabling operation for operator stations

In this area, you can disable the operation for an operator station or enable the operator station for the connected technologic block. In this case, the upper check box corresponds to the `In0` I/O and the lower check box to the `In15` I/O. Operation is only possible with the highest-level operator control permission (same as simulation) .

### (3) Display for the current operator station

The value of the current operator station is displayed as a gray dot in the corresponding line.

### (4) Display for operability

Display of the operability of the technologic block on the current operator station.

### (5) Navigation button for switching to the standard view of the technologic block

Use this navigation button to reach the standard view of the technologic block. The visibility of this navigation button depends on the configuration in the engineering system (ES).

Additional information is available in the section Opening additional faceplates (Page 234).

### (6) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

### 3.6.8.3 Block icon of OpStations

#### Block icons for OpStations

A variety of block icons are available with the following functions:

- Opening the faceplate

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
| Without icon | 1 | Default no block icon |
|  | 2 | The block icon is transparent, if operation on the operator station is possible |

**Note**

The block icon is the size of a field in the status bar of a technologic block icon and can be used as an add-on to the status bar. The layer of the block icon here should always be higher than the layer of the technologic block icon, otherwise the block icon may be hidden after an update.

# 3.7 OpTrig - Manipulating a digital value (1 pushbutton)

## 3.7.1 Description of OpTrig

### Object name (type + number) and family

Type + number: FB 1868

Family: Operate

### Area of application for OpTrig

The block is used for the following applications:

- Generation of a pulse signal (trigger)

### How it works

Operator control block is used to implement single pushbutton control (comparable with RESET pushbutton).

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section OpTrig I/Os (Page 474).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | Not used |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | Not used |
| 6 | `OnAct.Value` |
| 7 | `Out.Value` |
| 8 | `LiOpAct.Value` |
| 9 | `FbkIn.Value` |
| 10 - 31 | Not used |

**See also**

OpTrig functions (Page 471)

OpTrig messaging (Page 474)

OpTrig block diagram (Page 477)

OpTrig error handling (Page 473)

OpTrig modes (Page 471)

## 3.7.2 OpTrig modes

### OpTrig operating modes

The block can be operated using the following modes:

- On (Page 77)
- Out of service (Page 77)

### "On"

General information on the "On" mode is available in the section On (Page 77).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

**See also**

OpTrig block diagram (Page 477)

OpTrig I/Os (Page 474)

OpTrig messaging (Page 474)

OpTrig functions (Page 471)

OpTrig error handling (Page 473)

Description of OpTrig (Page 470)

## 3.7.3 OpTrig functions

### Functions of OpTrig

The functions for this block are listed below.

### Issuing trigger signal internally or externally

Use the parameter `LiOp` to define whether the trigger signal is to be output by interconnection or by the OS operator:

`LiOp` = 0: Trigger signal by OS operator (input parameter `InOp`)

`LiOp` = 1: Trigger signal via interconnection (input parameter `InLi`)

### Input parameter for feedback value

This block has a `FbkIn` input parameter for displaying a feedback value in the faceplate.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameter:

- `FbkIn.ST`
- `InLi.ST` (only if `Feature.Bit10` = 1)

### Considering bad quality of automatic commands or external values

If the `Feature` bit Considering bad quality of automatic commands or external values (Page 200) is set to 1 and the parameter `InLi.ST` has bad signal status (16#00 or 16#28), the block goes to the neutral position (`Out` = 0) in the "On" mode.

### Operator control permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can set the input parameter `In` |
| 5 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the section Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 1 | Reaction to the out of service mode (Page 188) |
| 10 | Considering bad quality of automatic commands or external values (Page 200) |
| 24 | Enabling local operator authorization (Page 169) |

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of OpTrig (Page 470)

OpTrig messaging (Page 474)

OpTrig I/Os (Page 474)

OpTrig block diagram (Page 477)

OpTrig error handling (Page 473)

OpTrig modes (Page 471)

## 3.7.4 OpTrig error handling

### OpTrig error handling

The block does not report any errors.

### See also

OpTrig block diagram (Page 477)
OpTrig I/Os (Page 474)
OpTrig messaging (Page 474)
OpTrig functions (Page 471)
OpTrig modes (Page 471)
Description of OpTrig (Page 470)

## 3.7.5 OpTrig messaging

### Messaging

This block does not offer messaging.

### See also

Description of OpTrig (Page 470)
OpTrig functions (Page 471)
OpTrig I/Os (Page 474)
OpTrig block diagram (Page 477)
OpTrig error handling (Page 473)
OpTrig modes (Page 471)

## 3.7.6 OpTrig I/Os

### OpTrig I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `BatchEn` | 1 = Enable allocation | BOOL | 0 |
| `BatchID` | Batch ID | DWORD | 16#00000000 |
| `BatchName` | Batch name | S7-String | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkTypeData* | This input can not be edit or linked and is for internal usage only! | | |
| EN | 1 = Called block will be processed | BOOL | 1 |
| FbkIn | Input for feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Feature | I/O for additional functions (Page 471) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| InLi | Interconnectable binary input | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| InOp* | Binary input for operator | BOOL | 0 |
| LiOp | Switchover of operating mode between:<br>1 = Interconnection<br>0 = Operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operating permission, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator control permissions (Page 471) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SelFp1 | Call a block saved in this parameter as additional faceplate (Page 234) in standard view | ANY | - |
| SelFp2 | Call a block saved in this parameter as additional faceplate (Page 234) in the preview | ANY | - |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Reserved | INT | -1 |
| LiOpAct | Operator/interconnection is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| Out | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status | Status word (Page 470) | DWORD | 16#00000000 |

**See also**

OpTrig messaging (Page 474)
OpTrig block diagram (Page 477)
OpTrig error handling (Page 473)
OpTrig modes (Page 471)

## 3.7.7 OpTrig block diagram

### OpTrig block diagram

A block diagram is not provided for this block.

**See also**

OpTrig I/Os (Page 474)
OpTrig messaging (Page 474)
OpTrig functions (Page 471)
OpTrig error handling (Page 473)
OpTrig modes (Page 471)
Description of OpTrig (Page 470)

## 3.7.8 Operator control and monitoring

### 3.7.8.1 OpTrig views

### Views of the OpTrig block

The block OpTrig provides the following views:

- OpTrig standard view (Page 478)
- OpTrig preview (Page 479)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for OpTrig (Page 480)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 3.7.8.2 OpTrig standard view

#### OpTrig standard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 77)
- Out of service (Page 77)

You can find additional information on this in the Switching operating states and operating modes (Page 289) section.

#### (2) Set

Clicking "Set", outputs a pulse signal with the length of the cycle time at the `Out` output.

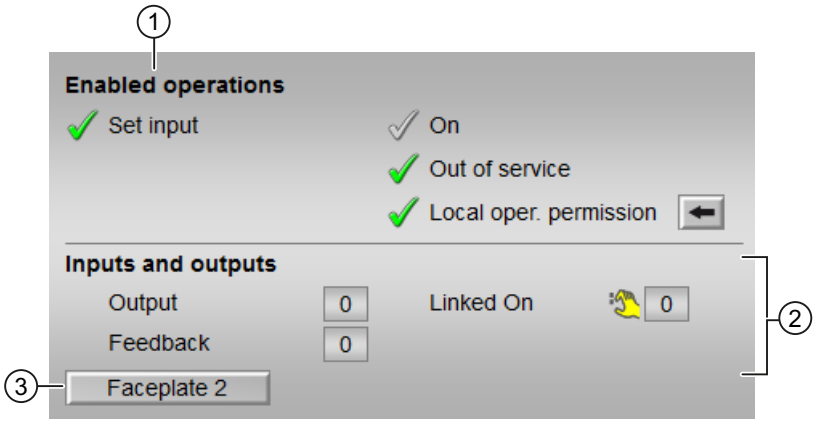
#### (3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the Engineering System (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Opening additional faceplates (Page 234) section.

### 3.7.8.3 OpTrig preview

#### Preview of OpTrig




#### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter.
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process.
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

The following enabled operations are shown here:

- "Set input": You can set the input.
- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. Additional information is available in the section Operator control permissions (Page 285).

#### (2) Display of current inputs and outputs

This area shows the most important parameters for this block with the current selection:

- "Output": 1 = Digital output value set
- "Feedback": 1 = Feedback set
- "Linked On": 1 = Linked input on

### (3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

Refer also to the Opening additional faceplates (Page 234) section for more on this.

## 3.7.8.4 Block icon for OpTrig

### Block icons for OpTrig

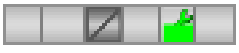
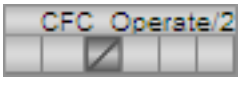
A variety of block icons are available with the following functions:

- Process tag type
- Operating modes
- Signal status, release for maintenance
- Memo display

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|   | 1 | |
|   | 2 | Block icon in the full display |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|   | 1 | Block icon in the full display |
|  | 2 | |
|   | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

# Condition Monitoring

# 4

## 4.1 PumpMon - Pump monitoring

### 4.1.1 Description of PumpMon

#### Object name (type + number) and family

Type + number: FB 1950

Family: CondMon

Number of process objects: 20 PO

#### Area of application of PumpMon

A lot of the products we use daily are produced in process engineering plants. Pumps are among the most important machines used in these plants. Around 20% of the world's electrical energy is consumed by pumps. If a pump fails, this can cause an entire plant to shut down and the resulting losses can quickly exceed the value of the pump many times over.

This is why the availability of pumps is a very important factor. Redundant systems and special monitoring systems are sometimes implemented to ensure the required availability, although they can be extremely expensive and complex.

The scope for saving energy, however, has yet to be fully leveraged. The PumpMon block for monitoring PCS 7 centrifugal pumps was developed as a cost-efficient monitoring solution and to leverage potential for saving energy.

The block is used to:

- Warn against potential damage to pumps under unfavorable operating conditions
- Provide early detection of imminent damage to pumps
- Optimize the pump design over the long term by means of statistical analyses of the operating data.

The block can be used for electric centrifugal pumps as well as for constant and variable-speed pumps.

#### Mode of operation

Any limit violations of the nominal pump operating range and deviations from the expected characteristic curve are visualized and reported to the operator and made available for further processing via the block outputs. All the values can of course be processed further with the usual PCS 7 tools (calculations, archiving, curve recording, alarm history, etc.).

The block itself is designed purely for diagnostic purposes and, as such, does not intervene directly in the operation of the pump. This means that it can be used (including as a retrofitted

device) without having to worry about affecting the process. If required, active intervention (e.g. to reduce the speed of the pump in response to imminent cavitation) can be undertaken by evaluating the block outputs.

The following views are provided:

- Delivery height characteristic: Displays the delivery height as a function of the flow rate at the current delivery height operating point of the pump
- Power and efficiency characteristic: Displays the pump power and the pump efficiency as a function of the flow rate at the current power and efficiency operating point of the pump
- NPSH characteristic: Displays the cavitation-relevant NPSH (Net Positive Suction Head) of the value as a function of the flow rate at the current NPSHa operating point of the pump
- Compares the recorded electrical performance, the calculated mechanical performance (via a motor model) and the hydraulic performance generated by the pump
- Provides statistical analyses of the operating conditions, taking into account the flow rate and cavitation reserve
- Various views for (one-time) input of pump and medium parameters

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

### Dynamic input values

As input data, the block requires the electrical power consumption of the motor, the flow rate, the differential pressure across the pump and a binary signal from which the operating state of the motor can be deduced. For cavitation monitoring, the pressure on the input side of the pump and the temperature of the pumped medium are also required. These signals must be provided by upstream blocks.

The following input signals must be interconnected:

- Flow rate of pumped medium (`Flow`)
- Inlet pressure, intake pressure (`P_In`) (if no cavitation monitoring is required, a constant value can be parameterized here)
- Output pressure, flow pressure (`P_Out`)
- Active electrical power (`PoElec`)
- Motor operating state binary signal (`Running`)

In addition for cavitation monitoring:

- Temperature of pumped medium (`Temp`)

The following input must also be interconnected for speed-controlled motors (`ConstSpd = FALSE` input):

- Speed (`Spd`)

If the converter calculates and supplies the mechanical power, the mechanical power must be interconnected at the following input:

- Mechanical power (PoMech) of converter (when ConvAct = TRUE)

### Static input values

The following data must be known and entered as static input data (most parameters can be entered via the faceplate):

- Density of the medium
- For cavitation monitoring, the vapor pressure curve of the medium (in the form of Antoine coefficients or via external calculation)
- Motor characteristic data (rated power, rated speed, for non-standardized motors: polynomial for the relation between mechanical shaft power and recorded electrical power)
- Pump characteristic data (diameter of intake and discharge stubs, characteristics for delivery height, mechanical power, efficiency, and NPSH value as a function of flow rate)

### Startup characteristics

Use Feature.Bit0 (Setting the startup characteristics) to define the startup characteristics of this block.

- **Bit = 0:** The most recently stored values are reset at startup.
- **Bit = 1**: The most recently used values are output to the following output parameters at startup:
  - siPoHydr
  - siPoMech
  - DelHi
  - NPSH
  - EtaOut
  - Devxxx
  - MinTolxxx
  - MaxTolxxx
  - Relxxx

  (xxx = DelHi, Power, Eta or Npsh)

Following startup, the messages are suppressed for the number of cycles assigned in the RunUpCycparameter.

**Note**

Additional information about Feature.Bit0 can be found in Set startup characteristics (Page 148).

### Status word assignment for the `Status1` parameter

For a description of the individual parameters, refer to the section "I/Os of PumpMon (Page 505)".

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | Not used |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `OnAct.Value` |
| 6 – 10 | Not used |
| 11 | `Run.Value` |
| 12 | `Bypass = 0`: No bypass set |
| 13 | `ConstSpd = 0`: Speed-controlled pump |
| 14 | `ConvAct` |
| 15 | `PolyAct` |
| 16 | Display of efficiency (`PolyAct = 0 AND ConvAct = 0`) |
| 17 | `Teach` |
| 18 – 24 | Not used |
| 25 | Idle time is active |
| 26 – 27 | Not used |
| 28 | Internal error |
| 29 – 31 | Not used |

### Status word assignment for `Status2` parameter

For a description of the individual parameters, refer to the section "I/Os of PumpMon (Page 505)".

| Status bit | Parameter |
|---|---|
| 0 | `MsgLock` |
| 1 | `DelHiAct.Value` |
| 2 | `PowerAct.Value` |
| 3 | `EtaAct.Value` |
| 4 | `NpshAct.Value` |
| 5 | `CaviAct.Value` |
| 6 | `DirAct.Value` |
| 7 | `BlkAct.Value` |
| 8 | `DryRunAct.Value` |
| 9 | `PoElecAH_Act.Value` |
| 10 | `PoElecWH_Act.Value` |
| 11 | `PoElecWL_Act.Value` |
| 12 | `PoElecAL_Act.Value` |
| 13 | `PoMechAH_Act.Value` |
| 14 | `PoMechWH_Act.Value` |

| Status bit | Parameter |
|---|---|
| 15 | PoMechWL_Act.Value |
| 16 | PoMechAL_Act.Value |
| 17 | PoHydrAH_Act.Value |
| 18 | PoHydrWH_Act.Value |
| 19 | PoHydrWL_Act.Value |
| 20 | PoHydrAL_Act.Value |
| 21 | DelHiEn |
| 22 | PowerEn |
| 23 | EtaEn |
| 24 | NpshEn |
| 25 | CaviEN |
| 26 | DirEn |
| 27 | BlkEn |
| 28 | DryRunEn |
| 29 | 1= Suppressed messages |
| 30 - 31 | Not used |

### Status word assignment for `Status3` parameter

For a description of the individual parameters, refer to the section "I/Os of PumpMon (Page 505)".

| Status bit | Parameter |
|---|---|
| 0 | PoElecAH_En |
| 1 | PoElecWH_En |
| 2 | PoElecWL_En |
| 3 | PoElecAL_En |
| 4 | PoMechAH_En |
| 5 | PoMechWH_En |
| 6 | PoMechWL_En |
| 7 | PoMechAL_En |
| 8 | PoHydrAH_En |
| 9 | PoHydrWH_En |
| 10 | PoHydrWL_En |
| 11 | PoHydrAL_En |
| 12 | DelHiMsgEn |
| 13 | PowerMsgEn |
| 14 | EtaMsgEn |
| 15 | NpshMsgEn |
| 16 | CaviMsgEn |
| 17 | DirMsgEn |
| 18 | BlkMsgEn |
| 19 | DryRunMsgEn |
| 20 – 31 | Not used |

### Status word assignment for the `Status4` parameter

For a description of the individual parameters, refer to the section "I/Os of PumpMon (Page 505)".

| Status bit | Parameter |
|---|---|
| 0 – 31 | Not used |

## 4.1.2 PumpMon modes

### PumpMon operating modes

The block can be operated using the following modes:

- On
- Out of service

### "On"

General information about the "On" operating mode can be found in the section "On (Page 77)" mode.

### "Out of service"

General information about the "Out of service" operating mode can be found in the section "Out of service (Page 77)" mode.

## 4.1.3 PumpMon functional description

In the CFC, PumpMon is supplied with current process values via the inputs `Flow` (flow rate), `P_In` (inlet pressure, pump intake pressure), `P_Out` (pump outlet pressure), `PoElec` (active electrical power of motor), and `Temp` (temperature of pumped medium). The `Spd` input is also required for speed-controlled drives.

The binary input `Running` signals whether or not the motor is running. A time period can be parameterized via the `IdleTime` input. When the motor is switched on, all messages are blocked until this time has elapsed. The default value is 5 [sec]. This suppression time is not effective when `IdleTime = 0`.

## Preprocessing

The process values are standardized to SI units via the *Offs and *Fact inputs. The normalized values are output again at the output. The status signals of the process values (quality code) are not evaluated but are instead output again at the output.

- siFlow.Value = FlowOffs + Flowfact * Flow.Value
  (normalize to $m^3/s$)
- siP_In.Value = P_Atmos + P_InOffs + P_InFact * P_In.Value
  (normalize to bar)
- siP_Out.Value = P_Atmos + P_OutOffs + P_OutFact * P_Out.Value
  (normalize to bar)
- siPoElec.Value = PoElecOffs + PoElecFact * PoElec.Value
  (normalize to kW)
- siSpd.Value = SpdOffs + SpdFact * Spd.Value
  (normalize to rpm)
- siTemp.Value = TempOffs + TempFact * Temp.Value
  (normalize to °C)

When the temperature value is normalized, note that the generally-applicable factors and offsets for temperature conversion need to be adjusted.

The following rules apply:

- TempFact = 1 / default factor
- TempOffs = - default offset / default factor

| Input variable | Default factor | **TempFact** | Default offset | **TempOffs** |
|---|---|---|---|---|
| °F | 1.8 | 0.555556 | 32.0 | -17.777779 |
| K | 1.0 | 1.0 | 273.15 | -273.15 |

P_Atmos is the air pressure that must be added to the measured pressures P_In and P_Out as an offset if the measured values only determine the overpressure.

## Calculating the mechanical power

The block covers 3 variants:

- Calculation for standard motors (default setting)
  (Parameter assignment: PolyAct = FALSE, ConvAct = FALSE)
  In this case the user only has to specify the efficiency and the rated power of the motor. The mechanical power is then calculated from the electrical power in the block using a generally applicable motor model.
- Calculation for other motors, e.g. canned motor pumps
  (Parameter assignment: PolyAct = TRUE, ConvAct = FALSE)
  In this case a unique motor characteristic is required; it may be requested e.g. from the pump manufacturer. It is used to calculate the mechanical power from the electrical power via a 4th-degree polynomial:

```
siPoMech.Value = CorrMech * (PolyData.K0
                                    + PolyData.K1  * siPoElec.Value
                                    + PolyData.K2  * siPoElec.Value ²
                                    + PolyData.K3  * siPoElec.Value ³
                                    + PolyData.K4  * siPoElec.Value ⁴)
```

In this case, the `Eta` parameter is not taken into account. The coefficients `PolyData.K0` to `PolyData.K4` must be supplied by the manufacturer of the motor/pump.

- The mechanical power is taken from a converter (or other external source)
  (Parameter assignment: `PolyAct = FALSE, ConvAct = TRUE`)
  If a converter outputs the mechanical power, this value is read by PumpMon via the `PoMech` input and, if necessary, normalized with the `PoMechOffs` and `PoMechFact` parameters.

### Slip correction

For electric motors there is slip (speed difference between stator rotary field and rotor) which affects the speed and the mechanical power.

If the value supplied externally (e.g. by the converter) is already slip-corrected for the speed, the block must not perform a second correction. In this case the input must be set to `SlipCorr = FALSE`, otherwise `SlipCorr = TRUE`.

Slip correction can only be carried out for variable-speed motors (`ConstSpd = FALSE`), as no speed is measured. For this reason, the setting for constant-speed motors is reset.

### Calculating the delivery height

The delivery height is calculated from the pressure difference and density of the medium in the pump.

The calculated delivery height (`DelHi.Value)` is limited to values `>= 0`.

### Calculating the hydraulic power

The hydraulic power is calculated from the normalized flow rate, the delivery height, and the density of the medium.

The calculated power (`siPoHydr.Value)` is limited to values `>= 0`.

### Calculating the NPSH value

NPSH stands for **N**et **P**ositive **S**uction **H**ead. The NPSH value, along with the flow rate and delivery height, is one of the key characteristic values of a pump. To ensure that the pump operates smoothly, the following must apply:

*NPSH (plant) > NPSH (pump)*

otherwise cavitation will occur.

The current NPSH value of the plant (NPSHa) is calculated from the vapor pressure, which can be calculated using an Antoine equation (`Antoine = TRUE`).

*Vapor pressure = factor * 10 (A – B/(Temp + C))*

The default values for water in the 1-100°C range are:

- `AntA = 8.07131`
- `AntB = 1730.63`
- `AntC = 233.426`
- `AntFact = 0.0013332` as conversion factor from [mmHg] (Torr) to [bar]

The parameters for other media must be adapted accordingly. The Antoine values for most materials are provided in the technical literature or on the Internet. If the integrated Antoine vapor pressure calculation is not used (`Antoine = FALSE`), the input value `P_Vapor` [bar] is used. In this case the vapor pressure must be calculated externally to the block.

The calculated NPSH value (`Npsh.Value)` is limited to values `>= 0`.

## Calculating the Eta efficiency

The efficiency is calculated from the ratio of the hydraulic power to the mechanical power.

The calculated efficiency (`EtaOut.Value)` is limited to values `>= 0`.

## Characteristics

A total of four characteristics are saved in the block. These are the characteristics for the delivery height, power, efficiency, and NPSH value. The characteristics each have 15 support values, whereby the following applies:

| | | | | |
|---|---|---|---|---|
| • x scale: | 0 <= `Flow1` <= ... | <= | `Flow15` | < = `FlowMax` |
| • y scale (delivery height): | `DelHi1` ... `DelHi15` | <= | `DelHiMax` | |
| • y scale (power): | `Power1` ... `Power15` | <= | `PowerMax` | |
| • y scale (efficiency): | `Eta1` ... `Eta15` | <= | `EtaMax` | |
| • x scale (NPSH): | 0 <= `FlowNp1` <= ... | <= | `FlowNp15` | < = `FlowMax` |
| • y scale (NPSH): | `Npsh1` ... `Npsh15` | <= | `NpshMax` | |

The x scale support values do not have to be equidistant.

The NPSH characteristic has separate support values for the flow rate (`FlowNp1`, ..., `FlowNp15`) because the characteristic cannot be determined by means of the teach function.

The values of the standardized flow rate in $m^3/h$ are used as input values for the x scale of the characteristics. The output values of the characteristics are determined by means of linear interpolation. The characteristic is not extrapolated beyond the operating range. If the flow rate value is less than the first support value, the first y value is output. If the flow rate value is greater than the 15th support value, the 15th y value is output.

The binary signals `DelHiAct.Value`, `PowerAct.Value`, `EtaAct.Value` and `NpshAct.Value` are set if the deviations from the characteristics, a settable time period

(xxx_A_DC and xxx_A_DG), are outside the configurable tolerances (xxxTol). The duration of the limit value violation is determined in separate counters.

### Special feature of variable-speed motors

The input ConstSpd = "variable" (FALSE) must be set for variable-speed motors. In addition, a value must be specified for the rated speed SpdRate and the current speed must be read via the Spd input.

### Blockage and dry running

The electrical power varies (non-linear) with the motor load (i.e. the flow rate).

During dry running, the intake-side valve is closed, the flow rate of the pumped medium is zero, and the electrical power value is reduced to a minimum value (<= PoDryRun).

When a blockage is present, the discharge-side valve is closed, the flow rate of the pumped medium is again zero, and the electrical power value is <= PoBlk.

The value during a blockage is higher than during dry running because the pump still contains pumped media, which is being circulated.

**Note**

The limit values PoBlk and PoDryRun only apply to operation under rated conditions. With variable-speed drives, the effective limits depend on the current speed.

### Reduction in delivery height, gas conveyance, cavitation, and incorrect direction of rotation

- A **reduction in delivery height** can have various causes:
  - Gas conveyance
  - Initial cavitation
  - Draw-in turbulence
  - Damage to the pump (wear)
  - Wrong direction of rotation (roughly half the delivery height with high power consumption)

  The block cannot differentiate this state without further information, so the general "Reduction in delivery height" message is triggered.
- **Gas conveyance** may occur if:
  - the delivery height is reduced, but the NPSH value remains in OK range.

  In this case, however, there may also be a reduction in delivery height as a result of wear, so no specified "Gas conveyance" message is triggered.
  It is possible to differentiate between wear and Gas conveyance by using the time profile of the reduction in delivery height. Wear generally occurs slowly (over weeks or months), whereas gas conveyance typically occurs within a short period (seconds to minutes).

- **Cavitation** is diagnosed when:
  - the NPSH value falls below the NPSH characteristic.
- **Incorrect direction of rotation** is diagnosed when:
  - the delivery height deviates significantly (current value <60% of the characteristic value) and
  - the power deviates slightly (current value > 80% of the characteristic value).

  The motor was connected incorrectly and rotates in the wrong direction.

### Determining the running time of the pump in different loading ranges

The flow rate range `0...FlowMax` is divided into 10 equal sub-ranges. Each of these sub-ranges is assigned a counter for the duration of stay:

- `Load1` = 0-10% of `FlowMax`
- `Load2` = 10-20 % of `FlowMax`
- `Load3` = 20-30% of `FlowMax`
- `Load4` = 30-40% of `FlowMax`
- `Load5` = 40-50% of `FlowMax`
- `Load6` = 50-60% of `FlowMax`
- `Load7` = 60-70 % of `FlowMax`
- `Load8` = 70-80% of `FlowMax`
- `Load9` = 80-90 % of `FlowMax`
- `Load10` = 90-100% of `FlowMax`

An additional counter is used to determine downtime.

- `Load0` is equal to the duration of standstill (`NOT Running.Value` or `PoElec.Value = 0`)

The `LoadRstOp` input is used to reset the counters and the overall counter `LoadTime` to zero and restart them. The counters are also reset when `FlowMax` is changed.

The `LoadTime` counter is incremented with the sampling time and specifies the monitoring duration in [h]. The individual counters are incremented on the basis of the flow rate value. They specify the running time of the pump for the sub-ranges and downtime in %.

### Determining the histogram for deviations of the NPSH value from the characteristic

To clarify the deviation from the NPSH value, 10 sub-ranges are created. A counter for the load period is assigned to each sub-range. The following rules apply:

- `DevNpsh1` = deviation `DevNpsh < -1 m` [%]
- `DevNpsh2` = deviation `DevNpsh` between `-1` and `-0.5 m` [%]
- `DevNpsh3` = deviation `DevNpsh` between `-0.5` and `0 m` [%]
- `DevNpsh4` = deviation `DevNpsh` between `0` and `+0.5 m` [%]

- `DevNpsh5` = deviation `DevNpsh` between `+0.5` and `+1 m` [%]
- `DevNpsh6` = deviation `DevNpsh` between `+1` and `+1.5 m` [%]
- `DevNpsh7` = deviation `DevNpsh` between `+1.5` and `+2 m` [%]
- `DevNpsh8` = deviation `DevNpsh` between `+2` and `+2.5 m` [%]
- `DevNpsh9` = deviation `DevNpsh` between `+2.5` and `+3 m` [%]
- `DevNPSH10` = deviation `DevNpsh > +3 m` [%]

An additional counter is used to determine downtime.

- `DevNpsh0` is equal to the duration of standstill (`NOT Running.Value` or `PoElec.Value = 0`)

The `LoadRstOp` input is used to reset the counters and the overall counter `LoadTime` to zero and restart them. The counters are also reset when `FlowMax` is changed.

The parameter (`LoadTime`) is incremented with the sampling time and specifies the pump monitoring duration in [h]. The individual counters are incremented on the basis of the NPSH value. They specify the running time of the pump for the sub-ranges and the downtime as a % of the monitoring period.

The values are displayed in a histogram on the faceplate and indicate that cavitation has occurred and therefore potential reductions in the life of the pump and a potential need for maintenance.

## Teach function

The characteristics in the pump documentation should normally be used. If the documentation is not available or the current status of the pump is to be used as a reference, the individual pump operating points can also be approached manually and the value determined here for the flow rate, delivery height, power, and efficiency can be used as support values for the characteristic values. The teach function can be used here as a tool for entering the support values for the characteristics. The values calculated for the delivery height, power, and efficiency are used as support values in the characteristics. The values can then be corrected manually if required.

Messages are suppressed while the teach function (`Teach = TRUE`) is active.

**Note**

The above condition of increasing values for the flow rate
(0 <= `Flow1` <= ... <= `Flow15` <= `FlowMax`) must also be met in this case.

**Note**

The teach function should only be used under rated conditions.

**Note**

The NPSH characteristic cannot be determined via the teach function. (This would also not be useful, as the pump would be damaged.) The data on the pump data sheet must be used for this purpose.

### Status signals

Two additional status signals are used to signal the status of the pump:

- **Motor running:** `Run.Value` (`= Running.Value`) and `Status1.Bit11`
- **Motor ramping up:** `Status1.Bit25`
  (Motor is running, but `IdleTime` has not expired)

### Bypass

With a bypass, an additional duct is used to convey pumped media from the discharge side back to the intake side of the pump. This is used to protect the pump and ensure that it is not damaged unintentionally.

A flow rate measurement will be inaccurate because the flow rate through the bypass is not measured. In this case, a substitute value for the flow rate is determined on the basis of the mechanical power.

The flow rate value in the characteristic that equates to the mechanical power is used here.

This calculation is only useful in the case of constant-speed pumps and when the power characteristic climbs continuously (no ambiguity).

**Note**

When the flow rate is determined on the basis of the power characteristic, the power characteristic is no longer monitored.

## 4.1.4 PumpMon Functions

The functions for this block are listed below.

### Alarm delays with two time values per limit pair

This block has the standard function "Alarm delays with two time values per limit pair (Page 227)".

### Limit monitoring of power values

This block uses the standard function "Limit monitoring of the process value (Page 93)" to monitor the following values for limit violations:

- Electrical power
- Mechanical power
- Hydraulic power

You can monitor the process value for the following high and low alarm and warning limits:

- `PoxxxxAH_Lim`: Limit value for high alarm
- `PoxxxxAL_Lim`: Limit value for low alarm

- `PoxxxxWH_Lim`: Limit value for high warning
- `PoxxxxWL_Lim`: Limit value for low warning

**Note**

Overshooting or undershooting of these limit values is not reported by the block.

If these states should nonetheless be reported, you must create these messages yourself with the available alarm outputs (`PoxxxxAH_Act`, `PoxxxxWH_Act`, `PoxxxxWL_Act`, and `PoxxxxAL_Act`) and an additional alarm block.

The electrical power is also monitored for the following limit values:

- Blockage
- Dry running

**Note**

Undershooting of these limit values is reported by the block.

### Tolerance monitoring of the characteristic curves

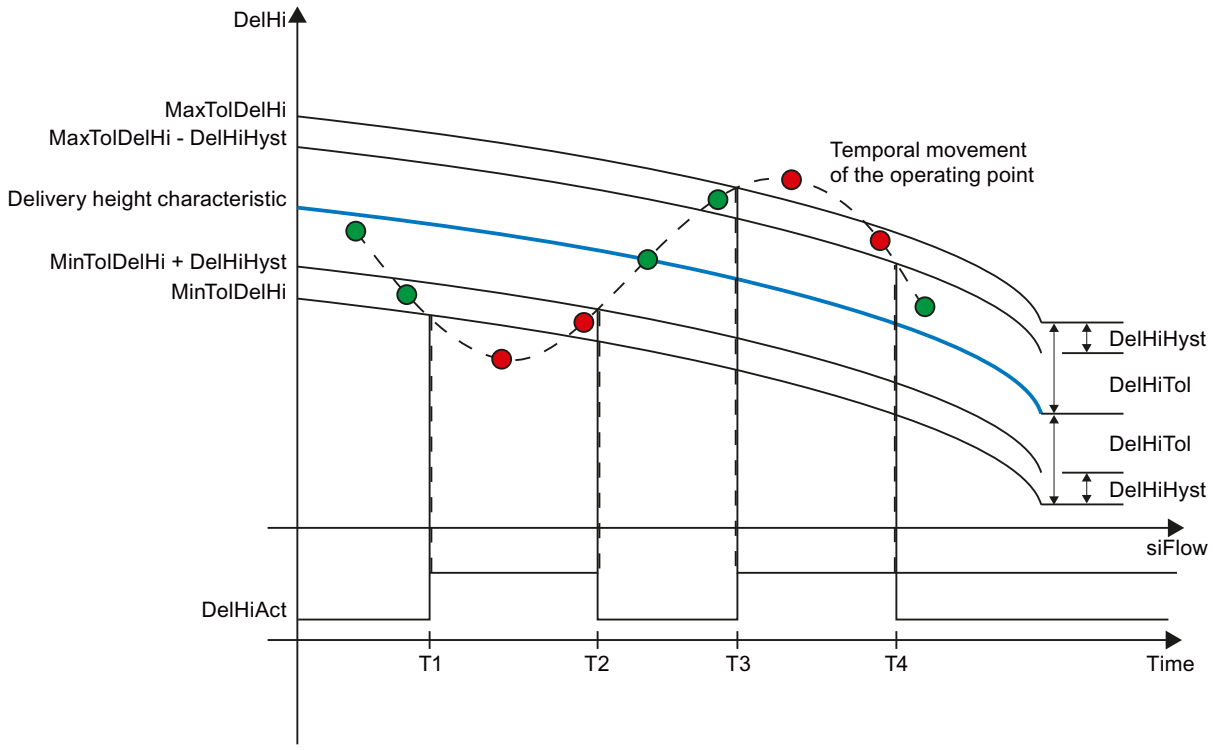
The block monitors the deviation of the applicable pump operating point from the relevant pump characteristic for settable tolerances.

The monitoring functionality is based on the standard "Limit monitoring of the process value (Page 93)" functionality.

The characteristic curve monitoring for

- Delivery height
- Performance
- Efficiency

Checks whether the operating point is within the tolerance range (see figure).

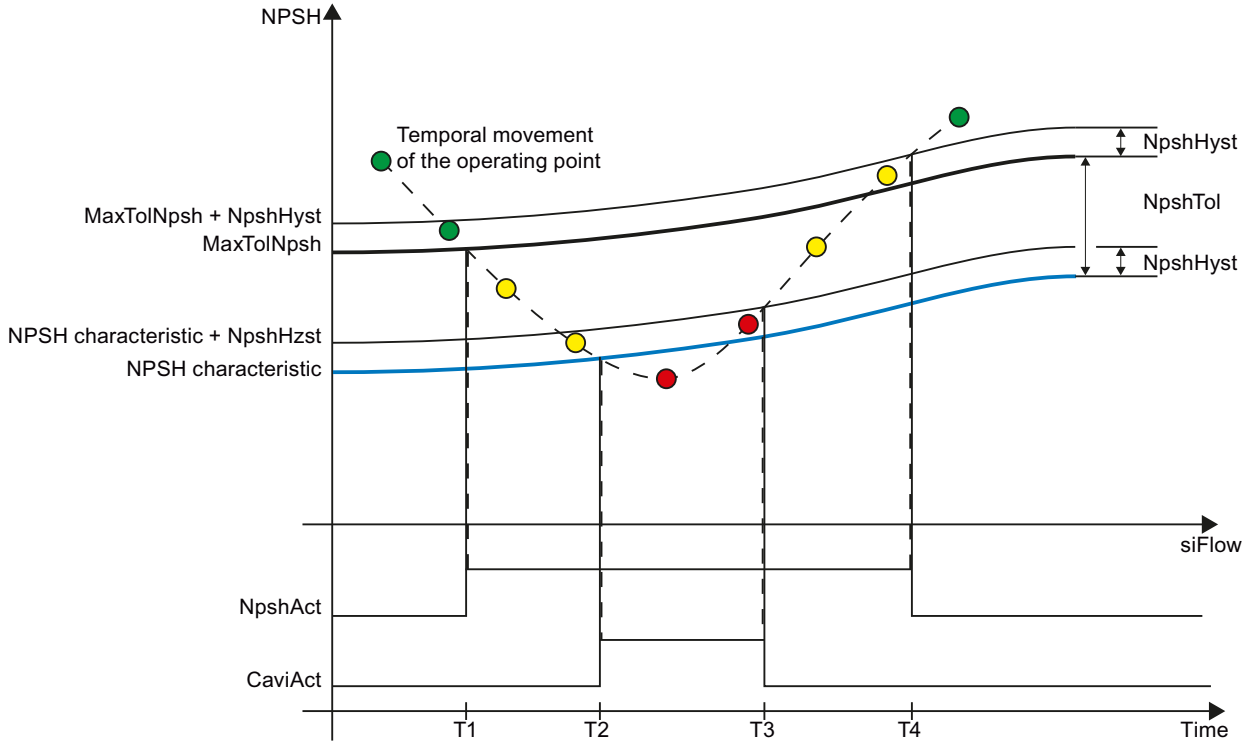



If the operating point leaves the tolerance range, the operating point in the faceplate changes its color from green to red and the corresponding message is triggered (`DelHiAct`, `PowerAct`, `EtaAct`).

Monitoring of the NPSH characteristic monitors the approach to the characteristic from above (see figure).

The following rules apply:

- If the operating point is in the tolerance range above the characteristic, the NPSH warning (`NpshAct`) is triggered and the operating point in the faceplate changes color from green to yellow.
- If the operating point is below the characteristic, the cavitation alarm (`CaviAct`) is triggered and the operating point in the faceplate changes color from yellow to red.




## Suppressing messages using the `MsgLock` parameter

This block has the standard function "Suppressing messages using the MsgLock parameter (Page 231)".

## Message suppression for speed-controlled pumps

In the case of speed-controlled pumps (`ConstSpd = FALSE`), the monitoring function for significant changes in speed for the time `IdleTime` in the block is suppressed. A significant change in speed in this case is a change of more than n% of one block pass to the next. n can be parameterized via the input parameter `DeltaSpd` (default = 5%).

**Note**

If in a specific application false alarms are still generated with this setting when the speed changes, the value can be reduced so that suppression applies even when the changes are smaller. However, it must be noted that the value must not be set too low, as otherwise statistical variations in the measured value or measurement noise would permanently suppress monitoring.

### Forming the signal status for blocks

This block has the standard function "Forming and outputting the signal status for technological blocks (Page 117)".

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `Flow.ST`
- `P_In.ST`
- `P_Out.ST`
- `PoElec.ST`
- `PoMech.ST`
- `Spd.ST`
- `Temp.ST`
- `DelHi.ST`
- `siPoHydr.ST`
- `EtaOut.ST`
- `Npsh.ST`

The following dependencies apply to the formation of the output-side signal status:

| Signal status | Dependencies |
|---|---|
| `siFlow.ST` | `Flow.ST` |
| `siP_In.ST` | `P_In.ST` |
| `siP_Out.ST` | `P_Out.ST` |
| `siSpd.ST` | `Spd.ST` |
| `siTemp.ST` | `Temp.ST` |
| `siPoElec.ST` | `PoElec.ST` |
| `siPoMech.ST` | • `PoElec.ST` (only when `ConvAct = FALSE`)<br>• `PoMech.ST` (only when `ConvAct = TRUE`)<br>• `Spd.ST` |
| `siPoHydr.ST` | • `Flow.ST`<br>• `P_In.ST`<br>• `P_Out.ST` |
| `DelHi.ST` | • `Flow.ST`<br>• `P_In.ST`<br>• `P_Out.ST`<br>• `Spd.ST` |
| `EtaOut.ST` | • `siPoMech.ST`<br>• `siPoHydr.ST` |
| `Npsh.ST` | • `Flow.ST`<br>• `P_In.ST`<br>• `Spd.ST`<br>• `Temp.ST` |

### Selecting a unit of measurement

This block provides the standard function "Selecting a unit of measure (Page 240)".

### Configurable reactions using the `Feature` parameter

An overview of all the reactions that are provided by the `Feature` parameter is available in section "Configurable response using the Feature I/O (Page 139)". The following modes of behavior are available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Setting the startup characteristics |
| 1 | Characteristics for the out of service mode |
| 3 | Disable limit act output when limit value message is disabled (Page 209) |
| 22 | Update acknowledgment and error status of the alarm call |
| 26 | Behavior of the switching points in "Out of service" mode |
| 28 | Disabling of switching points |
| 29 | Signaling limit violation |

### Operator control permissions

This block provides the standard function "Operator control permissions (Page 285)".

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 – 5 | Not used |
| 6 | 1 = Operator can reset the histogram |
| 7 – 9 | Not used |
| 10 | 1 = Operator can change the value for the rated power |
| 11 | 1 = Operator can change the value for the efficiency |
| 12 | 1 = Operator can change the value for the on delay |
| 13 | 1 = Operator can switch between constant and variable speed |
| 14 | 1 = Operator can enable slip correction |
| 15 | 1 = Operator can change the value for the rated speed |
| 16 | 1 = Operator can change the value for the diameter at the intake stub |
| 17 | 1 = Operator can change the value for the diameter at the discharge stub |
| 18 | 1 = Operator can change the value for the minimum flow rate |
| 19 | 1 = Operator can change the value for the optimum flow rate |
| 20 | 1 = Operator can enable the bypass |
| 21 | 1 = Operator can enable the Antoine calculation |
| 22 | 1 = Operator can change the value for the density |

| Bit | Function |
|---|---|
| 23 | 1 = Operator can change the characteristic curve values for the flow |
| 24 | 1 = Operator can change the characteristic curve values for the delivery height |
| 25 | 1 = Operator can change the characteristic curve values for the power |
| 26 | 1 = Operator can change the characteristic curve values for the efficiency |
| 27 | 1 = Operator can change the characteristic curve values for the flow rate (NPSH) |
| 28 | 1 = Operator can change the characteristic curve values for the NPSH value |
| 29 | 1 = Operator can enable the teach function |
| 30 – 31 | Not used |

The block has the following permissions for the `OS1Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can change the limit (`siPoElec`) for the high alarm |
| 1 | 1 = Operator can change the limit (`siPoElec`) for the high warning |
| 2 | 1 = Operator can change the limit (`siPoElec`) for the hysteresis |
| 3 | 1 = Operator can change the limit (`siPoElec`) for the low warning |
| 4 | 1 = Operator can change the limit (`siPoElec`) for the low alarm |
| 5 | 1 = Operator can change the limit (`siPoMech`) for the high alarm |
| 6 | 1 = Operator can change the limit (`siPoMech`) for the high warning |
| 7 | 1 = Operator can change the limit (`siPoMech`) for the hysteresis |
| 8 | 1 = Operator can change the limit (`siPoMech`) for the low warning |
| 9 | 1 = Operator can change the limit (`siPoMech`) for the low alarm |
| 10 | 1 = Operator can change the limit (`siPoHydr`) for the high alarm |
| 11 | 1 = Operator can change the limit (`siPoHydr`) for the high warning |
| 12 | 1 = Operator can change the limit (`siPoHydr`) for the hysteresis |
| 13 | 1 = Operator can change the limit (`siPoHydr`) for the low warning |
| 14 | 1 = Operator can change the limit (`siPoHydr`) for the low alarm |
| 15 | 1 = Operator can change the limit for the deviation from the delivery height characteristic curve |
| 16 | 1 = Operator can change the limit for the deviation from the power characteristic curve |
| 17 | 1 = Operator can change the limit for the deviation from the efficiency characteristic curve |
| 18 | 1 = Operator can change the limit for the deviation from the NPSH characteristic curve |
| 19 | 1 = Operator can change the limit for the blockage alarm |
| 20 | 1 = Operator can change the limit for the dry running alarm |
| 21 – 31 | Not used |

## Opening additional faceplates

You can open the standard view of the motor block (e.g. MotL) interconnected to the `Running` input parameter from the standard view via a button. The interconnection makes the button in the standard view visible.

### SIMATIC BATCH functionality

This block has the standard function "SIMATIC BATCH functionality (Page 73)".

## 4.1.5 PumpMon error handling

The block overwrites the following parameters with 0 if they were configured with a negative value:

- `DelHi_A_DC, DelHi_A_DG`
- `Power_A_DC, Power_A_DG`
- `Eta_A_DC, Eta_A_DG`
- `Npsh_W_DC, Npsh_W_DG`
- `Cavi_A_DC, Cavi_A_DG`
- `Blk_A_DC, Blk_A_DG`
- `DryRun_A_DC, DryRun_A_DG`
- `PoElec_A_DC, PoElec_A_DG`
- `PoElec_W_DC, PoElec_W_DG`
- `PoMech_A_DC, PoMech_A_DG`
- `PoMech_W_DC, PoMech_W_DG`
- `PoHydr_A_DC, PoHydr_A_DG`
- `PoHydr_W_DC, PoHydr_W_DG`
- `DelHiHyst`
- `PowerHyst`
- `EtaHyst`
- `NpshHyst`
- `BlkHyst`
- `DryRunHyst`
- `PoElecHyst`
- `PoMechHyst`
- `PoHydrHyst`
- `IdleTime`

### Parameter error

The output parameter `ER_PARA` is set to TRUE for one block cycle if one of the following conditions is met:

- `SampleTime <= 0.001`
- `PolyAct = TRUE` and `ConvAct = TRUE`

### Overview of error numbers

The `ErrorNum` I/O can be used to output various error numbers:

| Error ID | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |

## 4.1.6 PumpMon messaging

### Message behavior

The block uses an Alarm_8P block to generate messages. Messages can be triggered in the following cases:

- Limit violation of the electrical power (`siPoElec.Value`), the mechanical power (`siPoMech.Value`) and the hydraulic power (`siPoHydr.Value`) (e.g. overload state on overshoot of the high limit values for the electrical power). Cause of overload may be, for example: Increase of the viscosity of the pumped medium, valve opening is too large on the discharge side or bearing failure. Consequences of overload may include, for example: Overheating of the motor, more rapid wear.
- Deviation of the operating point from the configured characteristic (delivery height, power, Eta, and NPSH).
- Cavitation, dry running, blockage, and false alarm

### Process messages

| Message instance | Message ID | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 1 | High alarm | $$BlockComment$$<br>Reduction in delivery height: Delivery height (@8%.3f@m) deviates from characteristic (siFlow.Value =@4%f@m3/s) |
| | SIG 2 | High alarm | $$BlockComment$$<br>Mechan. power (@6%.3f@kW) deviates from characteristic<br>(siFlow.Value =@4%f@m3/s) |
| | SIG 3 | Low warning | $$BlockComment$$<br>Low NPSH value (@9%.2f@m)<br>(siFlow.Value =@4%f@m3/s) |
| | SIG 4 | High alarm | $$BlockComment$$<br>Efficiency (@10%.2f@%) deviates from characteristic<br>(siFlow.Value =@4%f@m3/s) |
| | SIG 5 | Low alarm | $$BlockComment$$<br>Cavitation<br>(DelHi.Value =@8%.2f@m, Npsh.Value =@9%.2f@m, siFlow.Value =@4%f@m3/s) |
| | SIG 6 | High alarm | $$BlockComment$$<br>Incorrect direction of rotation<br>(siFlow.Value =@4%f@m3/s) |
| | SIG 7 | High alarm | $$BlockComment$$<br>Blockage<br>(siPoElec.Value =@5%.3f@kW, siPoMech.Value =@6%.3f@kW, siPoHydr.Value =@7%.3f@kW, siFlow.Value =@4%f@m3/s) |
| | SIG 8 | High alarm | $$BlockComment$$<br>Dry running<br>(siPoElec.Value =@5%.3f@kW, siPoMech.Value =@6%.3f@kW, siPoHydr.Value =@7%.3f@kW, siFlow.Value =@4%f@m3/s) |

Explanation:

$$BlockComment$$: Content of the instance-specific block comment

### Associated values for the message instance `MsgEvId1`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | `siFlow.Value` |
| 5 | `siPoElec.Value` |
| 6 | `siPoMech.Value` |
| 7 | `siPoHydr.Value` |
| 8 | `DelHi.Value` |
| 9 | `Npsh.Value` |
| 10 | `EtaOut.Value` |

## 4.1.7 PumpMon I/Os

### I/Os of PumpMon

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `Accept` | 1 = Accept current values for teach function | BOOL | 0 |
| `AntA` | A value of the Antoine equation<br>For water from 1°C to 100°C `AntA = 8.07131` | REAL | 8.07131 |
| `AntB` | B value of the Antoine equation<br>For water from 1°C to 100°C `AntB = 1730.63` | REAL | 1730.63 |
| `AntC` | C value of the Antoine equation<br>For water from 1°C to 100°C `AntC = 233.426` | REAL | 233.426 |
| `AntFact` | Normalization factor of the vapor pressure in bar | REAL | 0.0013332 |
| `Antoine` | 1 = Antoine calculation is active | BOOL | 1 |
| `BatchEn` | 1 = Enable allocation | BOOL | 0 |
| `BatchID` | Batch ID | DWORD | 16#00000000 |
| `BatchName` | Batch name | S7-String | |
| `Blk_A_DC*` | Delay time for incoming blockage warnings [s] | REAL | 5.0 |
| `Blk_A_DG*` | Delay time for outgoing blockage warnings [s] | REAL | 5.0 |
| `BlkEn` | 1 = Enable blockage alarm limit | BOOL | 1 |
| `BlkHyst*` | Hysteresis for blockage alarm limit [kW] | REAL | 10.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkMsgEn | 1 = Enable message for blockage alarm | BOOL | 1 |
| Bypass | 1 = Bypass calculation is active | BOOL | 0 |
| Cavi_A_DC* | Delay time for incoming cavitation warnings [s] | REAL | 5.0 |
| Cavi_A_DG* | Delay time for outgoing cavitation warnings [s] | REAL | 5.0 |
| CaviEn | 1 = Enable cavitation alarm limit | BOOL | 1 |
| CaviMsgEn | 1 = Enable message for cavitation alarm | BOOL | 1 |
| ConstSpd | 1 = Motor is running at constant speed | BOOL | 1 |
| ConvAct | 1= Mechanical power is passed to PoMech<br>Used to calculate the mechanical shaft power (see PumpMon Functions (Page 495)). | BOOL | 0 |
| DelHi_A_DC* | Delay time for incoming alarm in the event of deviation from the delivery height characteristic [s] | REAL | 5.0 |
| DelHi_A_DG* | Delay time for outgoing alarm in the event of deviation from the delivery height characteristic [s] | REAL | 5.0 |
| DelHi1 | y value 1, delivery height [m] | REAL | 0.0 |
| DelHi2 | y value 2, delivery height [m] | REAL | 0.0 |
| DelHi3 | y value 3, delivery height [m] | REAL | 126.0 |
| DelHi4 | y value 4, delivery height [m] | REAL | 126.0 |
| DelHi5 | y value 5, delivery height [m] | REAL | 125.5 |
| DelHi6 | y value 6, delivery height [m] | REAL | 124.5 |
| DelHi7 | y value 7, delivery height [m] | REAL | 123.0 |
| DelHi8 | y value 8, delivery height [m] | REAL | 121.0 |
| DelHi9 | y value 9, delivery height [m] | REAL | 118.0 |
| DelHi10 | y value 10, delivery height [m] | REAL | 114.0 |
| DelHi11 | y value 11, delivery height [m] | REAL | 109.0 |
| DelHi12 | y value 12, delivery height [m] | REAL | 103.0 |
| DelHi13 | y value 13, delivery height [m] | REAL | 96.0 |
| DelHi14 | y value 14, delivery height [m] | REAL | 88.0 |
| DelHi15 | y value 15, delivery height [m] | REAL | 79.0 |
| DelHiCorr | Delivery height correction factor | REAL | 1.0 |
| DelHiEn | 1 = Enable alarm limit for deviation from the delivery height characteristic | BOOL | 1 |
| DelHiHyst* | Hysteresis for alarm limit in the event of deviation from the delivery height characteristic [%] | REAL | 0.1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| DelHiMax | Maximum delivery height value [m]<br>Used for the y axis of the delivery height characteristic | REAL | 140.0 |
| DelHiMsgEn | 1 = Enable message for alarm in the event of deviation from the delivery height characteristic | BOOL | 1 |
| DelHiTol | Tolerance for deviation from the delivery height characteristic [%] | REAL | 3.0 |
| DelHiUnit | Unit of measurement for the delivery height<br>(cannot be changed) | INT | 1010 |
| DeltaSpd | Speed change limit for message suppression in speed-controlled pumps [%]<br>In the event of a speed change of more than n% (n = DeltaSpd) from one block pass to the next, the monitoring functions are suppressed for the period IdleTime. | REAL | 5.0 |
| Density | Density of the pumped medium [kg/m³] | REAL | 1000.0 |
| DirEn | 1= Enable alarm limit for incorrect direction of rotation | BOOL | 1 |
| DirMsgEn | 1 = Enable message for alarm in the event of incorrect direction of rotation | BOOL | 1 |
| DryRun_A_DC* | Delay time for incoming dry running alarm [s] | REAL | 5.0 |
| DryRun_A_DG* | Delay time for outgoing dry run alarm [s] | REAL | 5.0 |
| DryRunEn | 1 = Enable dry running alarm limit | BOOL | 1 |
| DryRunHyst* | Hysteresis for dry running alarm limit [kW] | REAL | 10.0 |
| DryRunMsgEn | 1 = Enable message for dry running alarm | BOOL | 1 |
| EN | 1 = Called block is executed | BOOL | 1 |
| Eta | Efficiency of the motor (cos φ) [%] | REAL | 80.0 |
| Eta_A_DC* | Delay time for incoming alarm in the event of deviation from the efficiency characteristic [s] | REAL | 5.0 |
| Eta_A_DG* | Delay time for outgoing alarm in the event of deviation from the efficiency characteristic [s] | REAL | 5.0 |
| Eta1 | y value 1, efficiency [%] | REAL | 0.0 |
| Eta2 | y value 2, efficiency [%] | REAL | 0.0 |
| Eta3 | y value 3, efficiency [%] | REAL | 33.0 |
| Eta4 | y value 4, efficiency [%] | REAL | 41.0 |
| Eta5 | y value 5, efficiency [%] | REAL | 48.5 |
| Eta6 | y value 6, efficiency [%] | REAL | 54.5 |
| Eta7 | y value 7, efficiency [%] | REAL | 60.0 |
| Eta8 | y value 8, efficiency [%] | REAL | 64.5 |
| Eta9 | y value 9, efficiency [%] | REAL | 68.0 |
| Eta10 | y value 10, efficiency [%] | REAL | 70.5 |
| Eta11 | y value 11, efficiency [%] | REAL | 71.5 |
| Eta12 | y value 12, efficiency [%] | REAL | 71.5 |
| Eta13 | y value 13, efficiency [%] | REAL | 70.0 |
| Eta14 | y value 14, efficiency [%] | REAL | 67.0 |
| Eta15 | y value 15, efficiency [%] | REAL | 63.0 |
| EtaCorr | Efficiency correction factor | REAL | 1.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| EtaEn | 1 = Enable message for alarm in the event of deviation from the efficiency characteristic | BOOL | 1 |
| EtaHyst* | Hysteresis for alarm limit in the event of deviation from the efficiency characteristic [%] | REAL | 0.1 |
| EtaMax | Maximum value of efficiency [%]<br>Used for the y axis of the efficiency characteristic | REAL | 100.0 |
| EtaMsgEn | 1 = Enable message for alarm in the event of deviation from the efficiency characteristic | BOOL | 1 |
| EtaTol | Tolerance for deviation from the efficiency characteristic [%] | REAL | 3.0 |
| EtaUnit | Unit of measurement for efficiency (cannot be changed) | INT | 1342 |
| Feature | I/O for additional functions (see PumpMon Functions (Page 495)) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| Flow | Interconnectable process value for the flow rate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Flow1 | x value 1, flow [m³/h] | REAL | 100.0 |
| Flow2 | x value 2, flow [m³/h] | REAL | 200.0 |
| Flow3 | x value 3, flow [m³/h] | REAL | 300.0 |
| Flow4 | x value 4, flow [m³/h] | REAL | 400.0 |
| Flow5 | x value 5, flow [m³/h] | REAL | 500.0 |
| Flow6 | x value 6, flow [m³/h] | REAL | 600.0 |
| Flow7 | x value 7, flow [m³/h] | REAL | 700.0 |
| Flow8 | x value 8, flow [m³/h] | REAL | 800.0 |
| Flow9 | x value 9, flow [m³/h] | REAL | 900.0 |
| Flow10 | x value 10, flow [m³/h] | REAL | 1000.0 |
| Flow11 | x value 11, flow [m³/h] | REAL | 1100.0 |
| Flow12 | x value 12, flow [m³/h] | REAL | 1200.0 |
| Flow13 | x value 13, flow [m³/h] | REAL | 1300.0 |
| Flow14 | x value 14, flow [m³/h] | REAL | 1400.0 |
| Flow15 | x value 15, flow [m³/h] | REAL | 1500.0 |
| FlowFact | Normalization factor for the flow rate in m³/s | REAL | 2.7777e-4 |
| FlowMax | Maximum value of the flow rate [m³/h]<br>Use for the x axis of the delivery height, power, and efficiency characteristic | REAL | 1500.0 |
| FlowMin | Minimum value of the flow rate [m³/h]<br>Used for display on the characteristics | REAL | 700.0 |
| FlowNp1 | x value 1, NPSH flow [m³/h] | REAL | 100.0 |
| FlowNp2 | x value 2, NPSH flow [m³/h] | REAL | 200.0 |
| FlowNp3 | x value 3, NPSH flow [m³/h] | REAL | 300.0 |
| FlowNp4 | x value 4, NPSH flow [m³/h] | REAL | 400.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| FlowNp5 | x value 5, NPSH flow [m³/h] | REAL | 500.0 |
| FlowNp6 | x value 6, NPSH flow [m³/h] | REAL | 600.0 |
| FlowNp7 | x value 7, NPSH flow [m³/h] | REAL | 700.0 |
| FlowNp8 | x value 8, NPSH flow [m³/h] | REAL | 800.0 |
| FlowNp9 | x value 9, NPSH flow [m³/h] | REAL | 900.0 |
| FlowNp10 | x value 10, NPSH flow [m³/h] | REAL | 1000.0 |
| FlowNp11 | x value 11, NPSH flow [m³/h] | REAL | 1100.0 |
| FlowNp12 | x value 12, NPSH flow [m³/h] | REAL | 1200.0 |
| FlowNp13 | x value 13, NPSH flow [m³/h] | REAL | 1300.0 |
| FlowNp14 | x value 14, NPSH flow [m³/h] | REAL | 1400.0 |
| FlowNp15 | x value 15, NPSH flow [m³/h] | REAL | 1500.0 |
| FlowOffs | Normalization offset of the flow rate in m³/s | REAL | 0.0 |
| FlowOpt | Optimum flow rate [m³/h]<br>Used for display on the characteristics | REAL | 1200.0 |
| FlowUnit | Unit of measure for flow | INT | 0 |
| H_Geo | Geometric delivery height in [m]<br>Provides the difference in height between the intake and discharge stubs. | REAL | 0.0 |
| IdleTime* | Delay time [s] | REAL | 5.0 |
| InSocket | Discharge stub diameter [mm] | REAL | 300.0 |
| LineFreq | Line frequency [Hz]<br>Required for slip correction. | REAL | 50.0 |
| LoadRstOp* | 1 = Reset histogram data | BOOL | 0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section "Suppressing messages using the MsgLock parameter" (see "PCS 7 Advanced Process Library" manual). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Npsh_W_DC* | Delay time for incoming alarm in the event of deviation from the NPSH characteristic [s] | REAL | 5.0 |
| Npsh_W_DG* | Delay time for outgoing alarm in the event of deviation from the NPSH characteristic [s] | REAL | 5.0 |
| Npsh1 | y value 1, NPSH [m] | REAL | 0.00 |
| Npsh2 | y value 2, NPSH [m] | REAL | 0.00 |
| Npsh3 | y value 3, NPSH [m] | REAL | 0.95 |
| Npsh4 | y value 4, NPSH [m] | REAL | 1.00 |
| Npsh5 | y value 5, NPSH [m] | REAL | 1.07 |
| Npsh6 | y value 6, NPSH [m] | REAL | 1.17 |
| Npsh7 | y value 7, NPSH [m] | REAL | 1.29 |
| Npsh8 | y value 8, NPSH [m] | REAL | 1.44 |
| Npsh9 | y value 9, NPSH [m] | REAL | 1.62 |
| Npsh10 | y value 10, NPSH [m] | REAL | 1.83 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Npsh11 | y value 11, NPSH [m] | REAL | 2.07 |
| Npsh12 | y value 12, NPSH [m] | REAL | 2.34 |
| Npsh13 | y value 13, NPSH [m] | REAL | 2.64 |
| Npsh14 | y value 14, NPSH [m] | REAL | 2.96 |
| Npsh15 | y value 15, NPSH [m] | REAL | 3.30 |
| NpshCorr | NPSH correction factor | REAL | 1.0 |
| NpshEn | 1 = Enable message for alarm in the event of deviation from the NPSH characteristic | BOOL | 1 |
| NpshHyst* | Hysteresis for alarm limit in the event of deviation from the NPSH characteristic [m] | REAL | 0.1 |
| NpshMax | Maximum value of NPSH [m]<br>Used for the y axis of the NPSH characteristic | REAL | 3.5 |
| NpshMsgEn | 1 = Enable message for alarm in the event of deviation from the NPSH characteristic | BOOL | 1 |
| NpshTol | Tolerance for deviation from the NPSH characteristic [m] | REAL | 0.5 |
| NpshUnit | Unit of measurement for the NPSH value (cannot be changed) | INT | 1010 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator input | BOOL | 0 |
| OS_Perm | I/O for operator control permissions (see PumpMon Functions (Page 495)) | STRUCT<br>• Bit0: BOOL<br>• Bit10: BOOL<br>• Bit31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| OS1Perm | I/O for operator control permissions (see PumpMon Functions (Page 495)) | STRUCT<br>• Bit0: BOOL<br>• Bit10: BOOL<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| P_Atmos | Air pressure [bar] | REAL | 1.0 |
| P_In | Interconnectable process value of the intake pressure | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| P_InFact | Normalization factor of the intake pressure in bar | REAL | 1.0 |
| P_InOffs | Normalization offset of the intake pressure in bar | REAL | 0.0 |
| P_InUnit | Unit of measure for the input pressure | INT | 0 |
| P_Out | Interconnectable process value of the flow pressure | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| P_OutFact | Normalization factor of the flow pressure in bar | REAL | 1.0 |
| P_OutOffs | Normalization offset of the flow pressure in bar | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| P_OutUnit | Unit of measure for the flow pressure | INT | 0 |
| P_Vapor | Vapor pressure of the pumped medium [bar] | REAL | 0.0 |
| PoBlk | Limit alarm for the electrical power in the event of blockage [kW] | REAL | 200.0 |
| PoDryRun | Limit alarm for the electrical power in the event of dry running [kW] | REAL | 150.0 |
| PoElec | Interconnectable process value of the electrical power | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PoElec_A_DC* | Delay time for incoming electrical power alarms [s] | REAL | 5.0 |
| PoElec_A_DG* | Delay time for outgoing electrical power alarms [s] | REAL | 5.0 |
| PoElec_W_DC* | Delay time for incoming electrical power warnings [s] | REAL | 5.0 |
| PoElec_W_DG* | Delay time for outgoing electrical power warnings [s] | REAL | 5.0 |
| PoElecAH_En | 1 = Enable electrical power alarm limit (high) | BOOL | 1 |
| PoElecAH_Lim | Electrical power (high) alarm limit [kW] | REAL | 700.0 |
| PoElecAL_En | 1 = Enable electrical power alarm limit (low) | BOOL | 1 |
| PoElecAL_Lim | Electrical power (low) alarm limit [kW] | REAL | 210.0 |
| PoElecFact | Normalization factor of the electrical power in kW | REAL | 1.0 |
| PoElecHyst* | Hysteresis for the electrical power alarm and warning limits [kW] | REAL | 10.0 |
| PoElecOffs | Normalization offset of the electrical power in kW | REAL | 0.0 |
| PoElecUnit | Unit of measurement for electrical power | INT | 0 |
| PoElecWH_En | 1 = Enable electrical power warning limit (high) | BOOL | 1 |
| PoElecWH_Lim | Electrical power (high) warning limit | REAL | 650.0 |
| PoElecWL_En | 1 = Enable electrical power warning limit (low) | BOOL | 1 |
| PoElecWL_Lim | Electrical power (low) warning limit [kW] | REAL | 250.0 |
| PoHydr_A_DC* | Delay time for incoming hydraulic power alarms [s] | REAL | 5.0 |
| PoHydr_A_DG* | Delay time for outgoing hydraulic power alarms [s] | REAL | 5.0 |
| PoHydr_W_DC* | Delay time for incoming hydraulic power warnings [s] | REAL | 5.0 |
| PoHydr_W_DG* | Delay time for outgoing hydraulic power warnings [s] | REAL | 5.0 |
| PoHydrAH_En | 1 = Enable hydraulic power alarm limit (high) | BOOL | 1 |
| PoHydrAH_Lim | Hydraulic power (high) alarm limit [kW] | REAL | 500.0 |
| PoHydrAL_En | 1 = Enable hydraulic power alarm limit (low) | BOOL | 1 |
| PoHydrAL_Lim | Hydraulic power (low) alarm limit [kW] | REAL | 160.0 |
| PoHydrCorr | Hydraulic power correction factor | REAL | 1.0 |
| PoHydrHyst* | Hysteresis for the hydraulic power alarm and warning limits [kW] | REAL | 10.0 |
| PoHydrWH_En | 1 = Enable hydraulic power warning limit (high) | BOOL | 1 |
| PoHydrWH_Lim | Hydraulic power (high) warning limit [kW] | REAL | 450.0 |
| PoHydrWL_En | 1 = Enable hydraulic power warning limit (low) | BOOL | 1 |
| PoHydrWL_Lim | Hydraulic power (low) warning limit [kW] | REAL | 190.0 |
| PolePairs | Number of motor pole pairs for slip correction | REAL | 1.0 |
| PolyAct | 1 = Polynomial calculation active<br>Used to calculate the mechanical shaft power (see PumpMon Functions (Page 495)). | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PolyData | Polynomial coefficients<br>Used to calculate the mechanical shaft power (see PumpMon Functions (Page 495)). | STRUCT<br>• K0: REAL<br>• K1: REAL<br>• K2: REAL<br>• K3: REAL<br>• K4: REAL | -<br>• 0.0<br>• 0.0<br>• 0.0<br>• 0.0<br>• 0.0 |
| PoMech | Interconnectable process value of the mechanical power (e.g. from converter) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PoMech_A_DC* | Delay time for incoming mechanical power alarms [s] | REAL | 5.0 |
| PoMech_A_DG* | Delay time for outgoing mechanical power alarms [s] | REAL | 5.0 |
| PoMech_W_DC* | Delay time for incoming mechanical power warnings [s] | REAL | 5.0 |
| PoMech_W_DG* | Delay time for outgoing mechanical power warnings [s] | REAL | 5.0 |
| PoMechAH_En | 1 = Enable mechanical power alarm limit (high) | BOOL | 1 |
| PoMechAH_Lim | Mechanical power (high) alarm limit [kW] | REAL | 600.0 |
| PoMechAL_En | 1 = Enable mechanical power alarm limit (low) | BOOL | 1 |
| PoMechAL_Lim | Mechanical power (low) alarm limit [kW] | REAL | 180.0 |
| PoMechCorr | Mechanical power correction factor | REAL | 1.0 |
| PoMechFact | Mechanical power normalization factor in kW | REAL | 1.0 |
| PoMechHyst* | Hysteresis for the mechanical power alarm and warning limits [kW] | REAL | 10.0 |
| PoMechOffs | Normalization offset of mechanical power in kW | REAL | 0.0 |
| PoMechUnit | Unit of measurement for mechanical power | INT | 0 |
| PoMechWH_En | 1 = Enable mechanical power warning limit (high) | BOOL | 1 |
| PoMechWH_Lim | Mechanical power (high) warning limit [kW] | REAL | 550.0 |
| PoMechWL_En | 1 = Enable mechanical power warning limit (low) | BOOL | 1 |
| PoMechWL_Lim | Mechanical power (low) warning limit [kW] | REAL | 220.0 |
| PoRate | Rated power of the motor [kW] | REAL | 600.0 |
| PosTeach | Current position of the teach function | INT | 1 |
| Power_A_DC* | Delay time for incoming alarm in the event of deviation from the power characteristic [s] | REAL | 5.0 |
| Power_A_DG* | Delay time for outgoing alarm in the event of deviation from the power characteristic [s] | REAL | 5.0 |
| Power1 | y value 1, power [kW] | REAL | 0.0 |
| Power2 | y value 2, power [kW] | REAL | 0.0 |
| Power3 | y value 3, power [kW] | REAL | 318.0 |
| Power4 | y value 4, power [kW] | REAL | 339.0 |
| Power5 | y value 5, power [kW] | REAL | 360.0 |
| Power6 | y value 6, power [kW] | REAL | 380.0 |
| Power7 | y value 7, power [kW] | REAL | 399.0 |
| Power8 | y value 8, power [kW] | REAL | 417.0 |
| Power9 | y value 9, power [kW] | REAL | 434.0 |
| Power10 | y value 10, power [kW] | REAL | 450.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Power11 | y value 11, power [kW] | REAL | 465.0 |
| Power12 | y value 12, power [kW] | REAL | 480.0 |
| Power13 | y value 13, power [kW] | REAL | 495.0 |
| Power14 | y value 14, power [kW] | REAL | 509.0 |
| Power15 | y value 15, power [kW] | REAL | 522.0 |
| PowerEn | 1 = Enable message for alarm in the event of deviation from the power characteristic | BOOL | 1 |
| PowerHyst* | Hysteresis for alarm limit in the event of deviation from the power characteristic [%] | REAL | 0.1 |
| PowerMax | Maximum value of power [kW]<br>Used for the y axis of the power characteristic | REAL | 600.0 |
| PowerMsgEn | 1 = Enable message for alarm in the event of deviation from the power characteristic | BOOL | 1 |
| PowerOpScale | Limit for scale on the power bar graphs of the faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 750.0<br>• 0.0 |
| PowerTol | Tolerance for deviation from the power characteristic [%] | REAL | 3.0 |
| PresSocket | Discharge stub diameter [mm] | REAL | 300.0 |
| Running | 1 = Motor is running<br>The faceplate of the block interconnected at this block input can be called from the standard view | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| RunUpCyc | Number of cycles during startup, within these cycles messages are suppressed | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 1.0 |
| siFlowUnit | Unit of measurement for the normalized flow (cannot be changed) | INT | 1347 |
| siP_Unit | Unit of measurement for the normalized pressure (cannot be changed) | INT | 1137 |
| siPoUnit | Unit of measurement for the normalized power rating (cannot be changed) | INT | 1190 |
| siSpdUnit | Unit of measurement for the normalized speed (cannot be changed) | INT | 1083 |
| siTempUnit | Unit of measurement for the normalized temperature (cannot be changed) | INT | 1001 |
| SlipCorr* | 1 = Slip correction active | BOOL | 0 |
| Spd | Interconnectable process value for the speed | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SpdFact | Normalization factor of speed in rpm | REAL | 1.0 |
| SpdOffs | Normalization offset of speed in rpm | REAL | 0.0 |
| SpdRate | Rated speed [rpm] | REAL | 2900.0 |
| SpdUnit | Unit of measurement for the speed | INT | 0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| Teach | 1 = Teach function is active | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| `Temp` | Interconnectable process value for the temperature of the pumped medium | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `TempFact` | Normalization factor of the medium temperature in °C | REAL | 1.0 |
| `TempOffs` | Normalization offset of the medium temperature in °C | REAL | 0.0 |
| `TempUnit` | Unit of measurement for the temperature of the medium | INT | 0 |
| `TimeBase` | Time basis for the flow values of the characteristic (3600 = hour) [s]<br>The parameter cannot be changed. In the latest version, the characteristics can only be displayed with `TimeBase` = 3600 (equivalent to $m^3/h$). | REAL | 3600.0 |
| `UserStatus` | First freely assignable bits for use in OS | BYTE | 16#00 |
| `UserStatus2` | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `BlkAct` | 1 = Blockage alarm active<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `CaviAct` | 1 = Cavitation alarm active<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `DelHi` | Output delivery height [m] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `DelHiAct` | 1 = Deviation from the delivery height characteristic curve is too large<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `DevDelHi` | Deviation from the delivery height characteristic curve [m] | REAL | 0.0 |
| `DevEta` | Deviation from the efficiency characteristic curve [%] | REAL | 0.0 |
| `DevNpsh` | Deviation from the NPSH characteristic curve [m] | REAL | 0.0 |
| `DevNpsh0` | Downtime for NPSH histogram (`Not Running`) [%] | REAL | 0.0 |
| `DevNpsh1` | Running time of the NPSH value in the range < -1 m [%] | REAL | 0.0 |
| `DevNpsh2` | Running time of the NPSH value in the range -1 to -0.5 m [%] | REAL | 0.0 |
| `DevNpsh3` | Running time of the NPSH value in the range -0.5 to 0 m [%] | REAL | 0.0 |
| `DevNpsh4` | Running time of the NPSH value in the range 0 to 0.5 m [%] | REAL | 0.0 |
| `DevNpsh5` | Running time of the NPSH value in the range 0.5 to 1 m [%] | REAL | 0.0 |
| `DevNpsh6` | Running time of the NPSH value in the range 1 to 1.5 m [%] | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| DevNpsh7 | Running time of the NPSH value in the range 1.5 to 2 m [%] | REAL | 0.0 |
| DevNpsh8 | Running time of the NPSH value in the range 2 to 2.5 m [%] | REAL | 0.0 |
| DevNpsh9 | Running time of the NPSH value in the range 2.5 to 3 m [%] | REAL | 0.0 |
| DevNpsh10 | Running time of the NPSH value in the range > 3 m [%] | REAL | 0.0 |
| DevPower | Deviation from the power characteristic [kW] | REAL | 0.0 |
| DirAct | 1 = Alarm for incorrect direction of rotation active<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DryRunAct | 1 = Dry running alarm active<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ER_Para | 1 = Parameter error | BOOL | FALSE |
| ErrorNum | Output of the existing error number. For the error numbers that can be output by this block, see section PumpMon error handling (Page 502). | INT | -1 |
| EtaAct | 1 = Deviation from the efficiency characteristic is too large<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| EtaOut | Efficiency output [%] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Flow_h | Normalized flow rate for display in the faceplate [m³/h] | REAL | 0.0 |
| Load0 | Downtime for flow rate histogram (Not Running) [%] | REAL | 0.0 |
| Load1 | Running time of the pump in the range 0-10% FlowMax [%] | REAL | 0.0 |
| Load2 | Running time of the pump in the range 10-20% FlowMax [%] | REAL | 0.0 |
| Load3 | Running time of the pump in the range 20-30% FlowMax [%] | REAL | 0.0 |
| Load4 | Running time of the pump in the range 30-40% FlowMax [%] | REAL | 0.0 |
| Load5 | Running time of the pump in the range 40-50% FlowMax [%] | REAL | 0.0 |
| Load6 | Running time of the pump in the range 50-60% FlowMax [%] | REAL | 0.0 |
| Load7 | Running time of the pump in the range 60-70% FlowMax [%] | REAL | 0.0 |
| Load8 | Running time of the pump in the range 70-80% FlowMax [%] | REAL | 0.0 |
| Load9 | Running time of the pump in the range 80-90% FlowMax [%] | REAL | 0.0 |
| Load10 | Running time of the pump in the range 90-100% FlowMax [%] | REAL | 0.0 |
| LoadTime | Recording period of the histogram data [h] | REAL | 0.0 |
| MaxTolDelHi | Upper tolerance limit for deviation from the delivery height characteristic [m] | REAL | 0.0 |
| MaxTolEta | Upper tolerance limit for deviation from the efficiency characteristic [%] | REAL | 0.0 |
| MaxTolNpsh | Upper tolerance limit for approach to the NPSH characteristic [m] | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MaxTolPower | Upper tolerance limit for deviation from the power characteristic [kW] | REAL | 0.0 |
| MinTolDelHi | Lower tolerance limit for deviation from the delivery height characteristic [m] | REAL | 0.0 |
| MinTolEta | Lower tolerance limit for deviation from the efficiency characteristic [%] | REAL | 0.0 |
| MinTolPower | Lower tolerance limit for deviation from the power characteristic [kW] | REAL | 0.0 |
| MsgAckn1 | Message acknowledgment status 1 (output STATUS of first ALARM_8P) | WORD | 0 |
| MsgErr1 | 1 = Message error 1 (output ERROR of first ALARM_8P) | BOOL | FALSE |
| MsgStat1 | Message status 1 (output ERROR of first ALARM_8P) | WORD | 16#00 |
| Npsh | NPSH value output [m] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| NpshAct | 1 = NPSH warning active<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS1PermLog | Display of OS1Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1Perm | DWORD | 16#FFFFFFFF |
| PoElecAH_Act | 1 = Electrical power (high) alarm active<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PoElecAL_Act | 1 = Electrical power (low) alarm active<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PoElecWH_Act | 1 = Electrical power (high) warning active<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PoElecWL_Act | 1 = Electrical power (low) warning active<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>Value: BOOL<br>ST: BYTE | -<br>0<br>16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PoHydrAH_Act | 1 = Hydraulic power (high) alarm active<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PoHydrAL_Act | 1 = Hydraulic power (low) alarm active<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PoHydrWH_Act | 1 = Hydraulic power (high) warning active<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PoHydrWL_Act | 1 = Hydraulic power (low) warning active<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PoMechAH_Act | 1 = Mechanical power (high) alarm active<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PoMechAL_Act | 1 = Mechanical power (low) alarm active<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PoMechWH_Act | 1 = Mechanical power (high) warning active<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PoMechWL_Act | 1 = Mechanical power (low) warning active<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PowerAct | 1 = Deviation from the power characteristic is too large<br>You can change the reaction for this parameter by setting Feature Bit 28 (disabling operating points) and Feature Bit 29 (signaling limit violation) accordingly (see section PumpMon Functions (Page 495)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RelDelHi | Relative deviation from the delivery height characteristic [%] | REAL | 0.0 |
| RelEta | Relative deviation from the efficiency characteristic [%] | REAL | 0.0 |
| RelNpsh | Relative deviation from the NPSH characteristic [%] | REAL | 0.0 |
| RelPower | Relative deviation from the power characteristic [%] | REAL | 0.0 |
| Run | 1 = Motor is running | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| siFlow | Normalized flow rate output [$m^3/h$] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| siP_In | Normalized intake pressure output [bar] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| siP_Out | Normalized flow pressure output [bar] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| siP_Vapor | Vapor pressure of the pumped medium output [bar] | REAL | 0.0 |
| siPoElec | Normalized electrical power output [kW] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| siPoHydr | Normalized hydraulic power output [kW] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| siPoMech | Normalized mechanical power output [kW] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| siSpd | Normalized speed output [rpm] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| siTemp | Normalized temperature output [°C] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 | DWORD | 16#00000000 |
| Status2 | Status word2 | DWORD | 16#00000000 |
| Status3 | Status word3 | DWORD | 16#00000000 |
| Status4 | Status word4 | DWORD | 16#00000000 |
| SumMsgAct | 1 = Group signal is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### 4.1.8 PumpMon block diagram

A block diagram is not provided for this block.

### 4.1.9 PumpMon additional information

#### Relationship between measuring variables and diagnosable problems

| | Flow | Pressure differential across the pump | Active electrical power | Pressure upstream of the pump | Temperature of pumped medium | Density | Vapor pressure equation |
|---|---|---|---|---|---|---|---|
| Blockage | (X) [1] | | X[4] | | | | |
| Dry running | (X) [1] | | X[4] | | | | |
| Gas conveyance | (X) [2] | X | | | | X[6] | |
| Cavitation | X | | | X | X[5] | | X[7] |
| Wear | X | (X) [3] | X | | | | |
| Overload | | | X | | | | |
| Low pump efficiency | X | X | X | | | | |

Annotations:

- [1] Not strictly necessary, but useful for additional plausibility checks.
- [2] For correction purposes when the diameters of the intake and discharge stubs are different.
- [3] More apparent in the flow characteristic than in the power characteristic.
- [4] Used to calculate the mechanical power (more significant than the electrical power).
- [5] Insofar as this is not constant.
- [6] Insofar as this is not constant; can be used as auxiliary value for flow rate measurement.
- [7] Implemented in the block for water with a temperature of up to 100°C (Antoine equation); to be supplied by adjusted Antoine coefficients or by external calculation for other media.

## Diagnostics logic

The block features the following diagnostic functions:

- **Generating process alarms to warn operating personnel in unfavorable operating conditions**
  The following messages can be generated by the block supplied:
  - Limit violation for the three power values (e.g. electrical overload)

    **Note**

    No messages are generated by the block for this purpose. The three power values are only monitored for the limits and a limit violation is provided for processing to the block as binary output.

  - Deviation of the operating point from the delivery height characteristic (i.e. reduction in delivery height: indication of gas conveyance, cavitation, blockage, or dry running).
  - Deviation of the operating point from the power characteristic.
  - Low pump efficiency: determined by means of the deviation of the current pump efficiency (ratio of hydraulic power to mechanical power) from the efficiency characteristic.
  - Cavitation: determined by means of the calculated NPSHa value; early warning when an NPSH reserve is undershot vis-à-vis the NPSHr curve by x meters (default: x = 0.5m).
  - Gas conveyance: determined by means of the reduction in delivery height (> x% deviation from the flow characteristic; default: x = 3%) when cavitation is not present.
  - Blockage: determined on the basis of a limit value for the electrical power being undershot. Can also be detected when the flow rate value (if available) is (almost) zero.
  - Dry running: determined on the basis of a (second, lower) limit value for the electrical power being undershot. Can also be detected when the flow rate value (if available) is (almost) zero.
  - Wrong direction: (i.e. motor has been connected incorrectly and is rotating in the wrong direction): determined when the delivery height falls significantly (> 40%) but with only a slight deviation (< 20%) of the power characteristic.

  All limits are settable and all alarms can be suppressed.
- **Generating maintenance alarms to indicate advanced pump wear**
  This is not done directly from the block, but via a downstream AssetMon block (PCS 7 standard libraries). A range of different applications are possible here; a typical scenario would be as follows:
  Pump wear is indicated by a continuous reduction (over hours or days) in the delivery height (gas conveyance and cavitation can also have the same effect, although this should only be temporary). The "Deviation from flow characteristic" alarm could, therefore, be integrated chronologically by means of an operating hours counter (included in the standard scope of delivery of PCS 7) and a maintenance alarm ("Pump worn out?") triggered when a particular limit (e.g. three days) is reached.
  However, there is currently limited operating experience of how large a conformity error must be to indicate a pending pump failure.
  Another useful diagnostic alarm could be generated when a particular number of operating hours for the pump in cavitation is reached. For this purpose, a second operating hours counter would have to be started when NPSHa < NPSH characteristic. In this case, an alarm

("Pump damaged due to cavitation?") should be triggered after just a few hours in cavitation, although no generally-applicable guidelines have yet been defined for this.

- **Creating statistics (display of the pump's load profile) to analyze the suitability of the pump design**
  The distribution of flow values is displayed in the top diagram in the "Histogram view of PumpMon". Ideally, this distribution should reach its maximum close to the optimum operating point of the pump (vertical green line). It is generally presumed, however, that many pumps in the process industry are intentionally overdimensioned, although this means that energy is lost when the flow rate is controlled by means of throttle valves (which is the usual case). This diagram can help to choose a more suitable design when the pump is replaced – or to justify installing a converter (Siemens supplies an extensive range of converters).
  The distribution of the distances of the pump operating point from the NPSH curve (the "cavitation reserve") is displayed in the bottom diagram. Ideally, none of the values should fall below 0. If this continues to occur in the histogram, the data recorded in the control system can be analyzed to determine which (unexpected) boundary conditions have led to this. If needed, the operating time in cavitation can also be used for remaining life time calculations as operating experience increases.

## 4.1.10 Operator control and monitoring

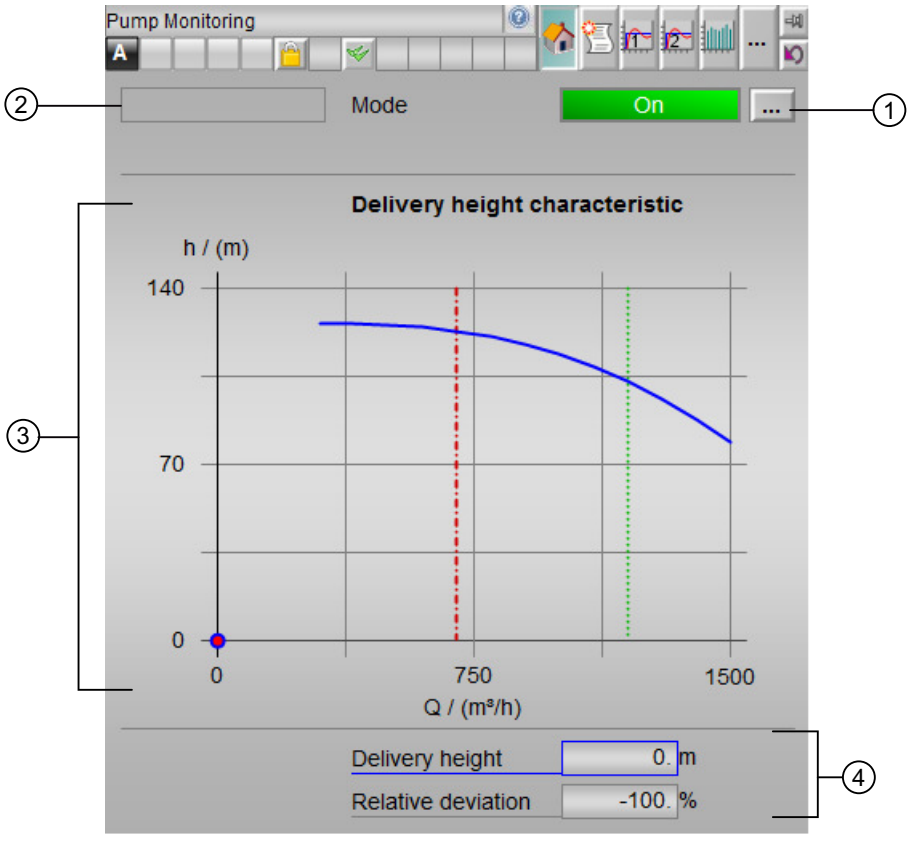
### 4.1.10.1 Views of PumpMon

The PumpMon block provides the following views:

- PumpMon standard view (Page 522) (delivery height characteristic)
- Alarm View (Page 336)
- Efficiency and power characteristic (Page 524)
- NPSH characteristic (Page 525)
- Histogram (Page 526)
- Parameter view 1 (Page 529)
- Parameter view 2 (Page 531)
- Parameter view 3 (Page 534)
- Preview (Page 535)
- View of power limits (Page 537)
- View of characteristic tolerances (Page 538)
- Trend view (Page 341)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for PumpMon (Page 539)

For general information on the faceplate and block icon, refer to the sections "Structure of the faceplate (Page 279)" and "Block icon structure (Page 260)".

### 4.1.10.2 PumpMon standard view (delivery height characteristic)




**(1) Display and switchover of operating mode**
This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On
- Out of service

**(2) Display area for block states**
This area provides additional information on the operating state of the block:

- "Delay"
- "Teach"

**(3) Jump button for switching to the standard view of any faceplate**
This display is only visible when the corresponding `Running` block input is connected. Use this jump button to reach the standard view of the interconnected block.

(4) Displays the delivery height/delivery capacity characteristic (delivery height)
The interpolation points for the characteristic curve are entered in the "Characteristic parameters" view.

If the distance of the current operating point exceeds a predefined tolerance (`DelHiTol`), the color of the operating point changes from green to red and a corresponding alarm is triggered (`DelHiAct.Value`).

The red line indicates the minimum permissible flow rate (FlowMin), while the green line indicates the optimum flow rate (FlowOpt).

The red line is only shown when FlowMin is > 5% of FlowMax.

The scaling of the X and Y axis is dynamic

- Y: 0 -> DelHiMax / 2 -> DelHiMax
- X: 0-> FlowMax / 2 -> FlowMax

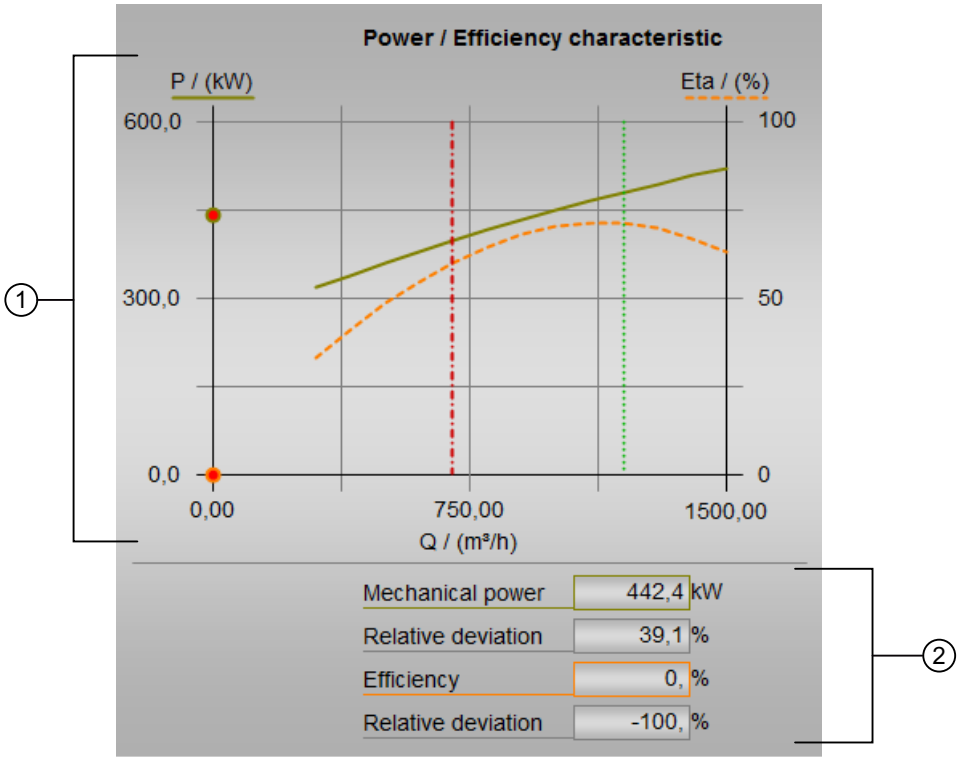
For ergonomic reasons as well as to optimize the computing time and refresh speed when the speed of variable-speed pumps changes (ConstSpd = 0), the characteristic is not re-calculated but, instead, the influence of the speed change is factored into the positions of the operating points. This occurs in accordance with the following dependencies:

```
siFlow_new(i)   = siFlow.Value(i)   * siSpd.Value  /  SpdRate
DelHi_new(i)    = DelHi.Value(i)    * (siSpd.Valu  /  SpdRate)2
                                      e
NPSH_new(i)     = Npsh.Value(i)     * (siSpd.Valu  /  SpdRate)2
                                      e
Power_new(i)    = Power.Value(i)    * (siSpd.Valu  /  SpdRate)3
                                      e
```

This re-scaling does not affect the original characteristic. However, the x and y axis scales would have to be permanently adjusted, which would lead to high load on the system and poor readability. The axis labels are therefore hidden instead. The positions of the lines for minimum ((FlowMin) and optimum (FlowOpt) flow rate are adjusted.

**(5) Display of the characteristic values**
This area shows the currently calculated delivery height (DelHi.Value) and the relative deviation (RelDelHi) from the characteristic. The values cannot be operated.

### 4.1.10.3 PumpMon efficiency and power characteristic view




**(1) Displays the power and efficiency characteristic with operating points**
This view contains the characteristics for power and efficiency.

Again, the color of the operating points changes to red when the distance between the operating point and the characteristic exceeds a predefined tolerance (`PowerTol` and `EtaTol`). An alarm is also triggered (`PowerAct.Value` and `EtaAct.Value`).

The scaling of the X and Y axes is dynamic:

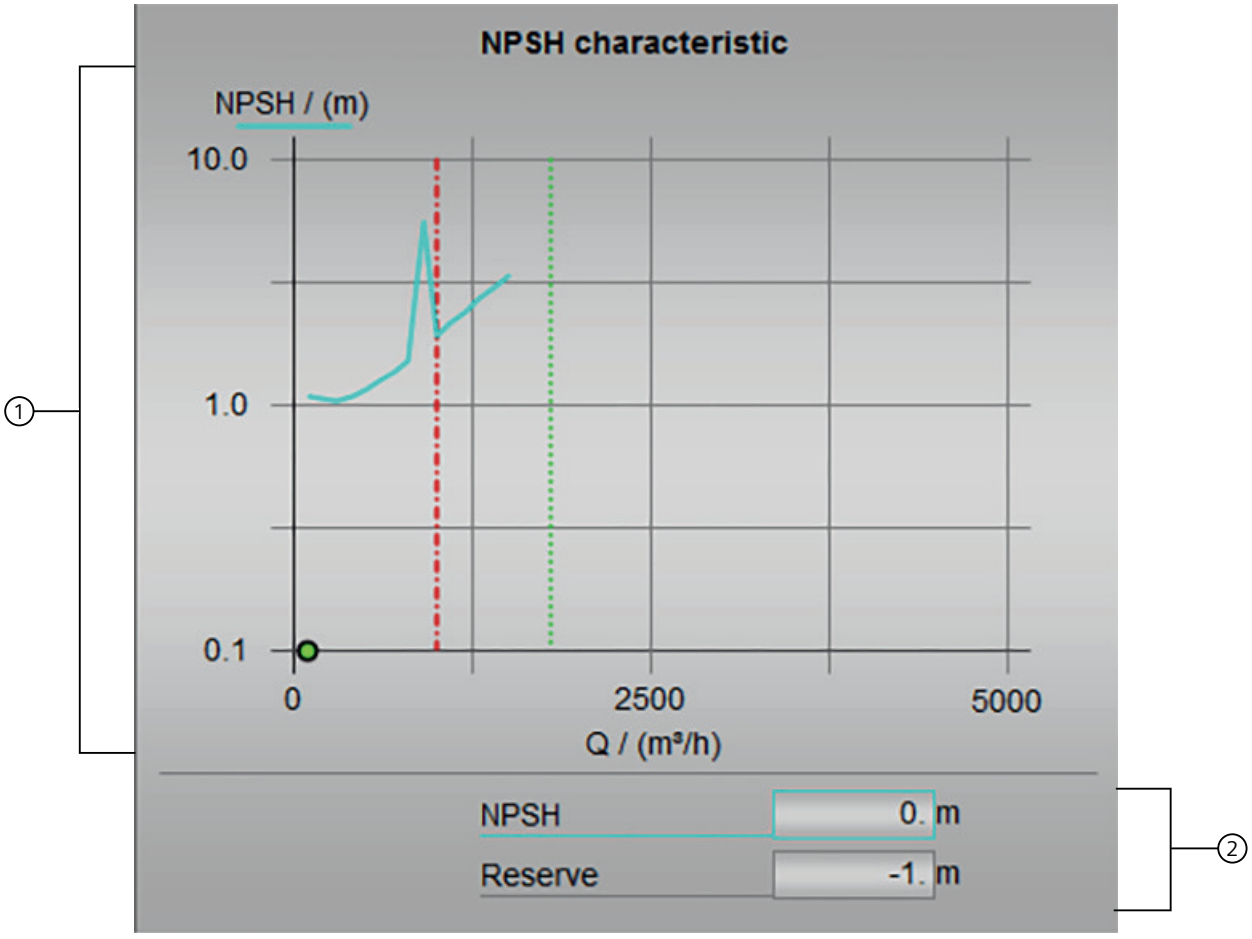
- **Y Power:** `0 -> PowerMax / 2 -> PowerMax`
- **Y Eta:** `0 -> EtaMax / 2 -> EtaMax`
- **X:** `0 -> FlowMax / 2 -> FlowMax`

The axis scale is displayed as per the behavior described above, dependent on the `ConstSpd` input parameter.

For more information. refer:Bar graph (Page 376)

**(2) Display of the characteristic values**
This area shows the currently calculated values for the mechanical power (`siPoMech.Value`), the efficiency (`EtaOut.Value`) and the relative deviations from the characteristics (`RelPower`) and (`RelEta`). The values cannot be operated.

### 4.1.10.4 PumpMon NPSH characteristic view




**(1) Display of the NPSH characteristic with operating point**
The NPSH value is displayed semi-logarithmically. The unit of measurement is [m]. A warning (`NpshAct.Value`) is triggered in this case when the operating point of the characteristic approaches a programmable value (`NpshTol` default = `+0.5 m`) from above. The operating point then changes color from green to yellow.

If the operating point falls below the characteristic, an alarm (`CaviAct.Value`) is triggered. The operating point changes color from yellow to red.

The minimum (`FlowMin`) and optimum (`FlowOpt`) flow rate is indicated in the same way as in the other characteristics.

The scaling of the X and Y axes is dynamic:

- **Y:**`0.1 -> Exp((Log(NpshMax)+Log(0.1)) / 2) -> NpshMax`
- **X:** `0 -> FlowMax / 2 -> FlowMax`

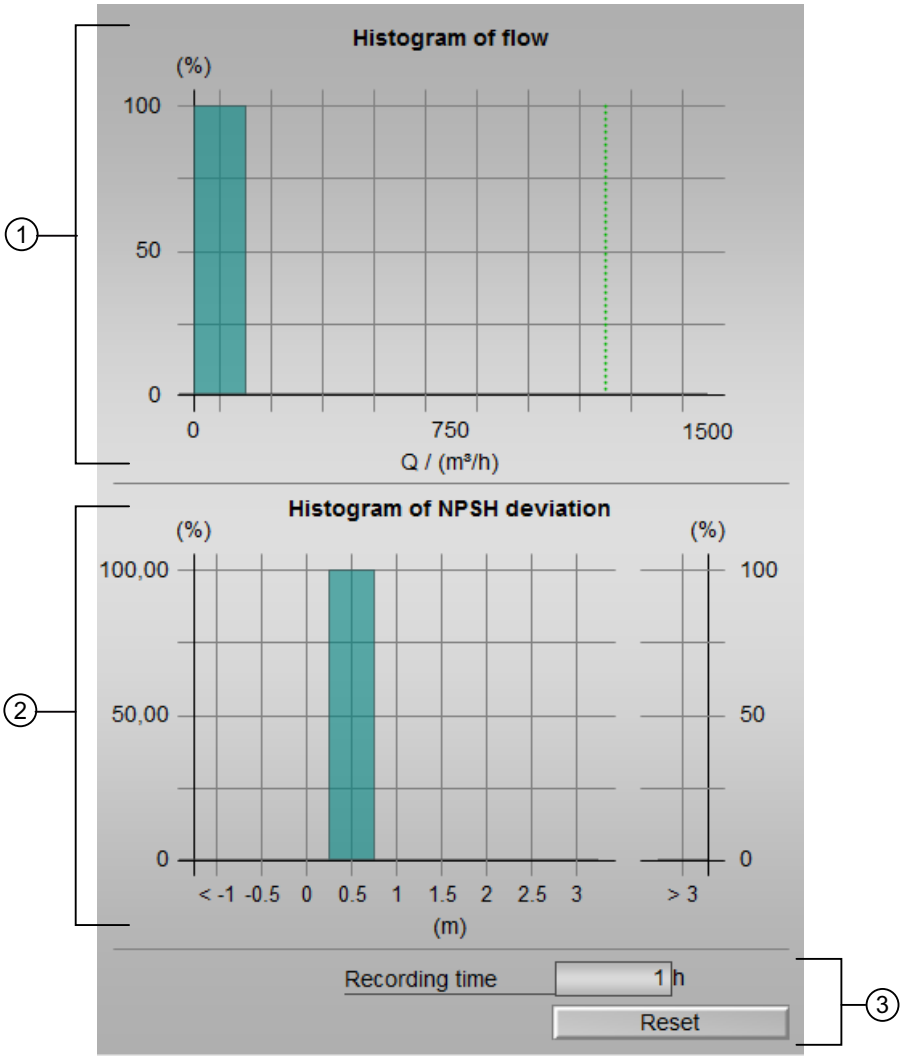
The axis scale is displayed as per the behavior described above, dependent on the `ConstSpd` input parameter.

For more information, refer:Bar graph (Page 376)

**(2) Display of the characteristic values**
This area shows the current NPSH value (`Npsh.Value`) and the relative deviation (`RelNpsh`) from the characteristic. The values cannot be operated.

### 4.1.10.5 PumpMon histogram view

These histograms are used to evaluate the "load profile" to which the corresponding pump has been exposed.




**(1) Flow rate histogram**
The histogram consists of a total of 11 bars. The first bar on the left (purple) indicates the idle times (`Load0`). The other 10 bars (cyan) divide the flow rate range from 0 to `FlowMax` into 10 equal sub-ranges. Each of these sub-ranges is assigned a counter for the duration of stay. The following rules apply:

- `Load1` = 0-10% of `FlowMax`
- `Load2` = 10-20% of `FlowMax`

- Load3 = 20-30% of FlowMax
- Load4 = 30-40% of FlowMax
- Load5 = 40-50% of FlowMax
- Load6 = 50-60% of FlowMax
- Load7 = 60-70% of FlowMax
- Load8 = 70-80% of FlowMax
- Load9 = 80-90% of FlowMax
- Load10 = 90-100% of FlowMax

The green line indicates the optimum flow rate (FlowOpt).

The scaling of the X and Y axes is dynamic:

- **Y:** 0 -> 50 -> 100%
- **X:** 0 -> FlowMax / 2 -> FlowMax

**(2) Histogram for deviation from the NPSH value**

To clarify the deviation from the NPSH value, 10 sub-ranges are created. A counter for the load period is assigned to each sub-range. The following rules apply:

- DevNpsh1 = deviation DevNpsh < -1 [m]
- DevNpsh2 = deviation DevNpsh between -1 and -0.5 [m]
- DevNpsh3 = deviation DevNpsh between -0.5 and 0 [m]
- DevNpsh4 = deviation DevNpsh between 0 and +0.5 [m]
- DevNpsh5 = deviation DevNpsh between +0.5 and +1 [m]
- DevNpsh6 = deviation DevNpsh between +1 and +1.5 [m]
- DevNpsh7 = deviation DevNpsh between +1.5 and +2 [m]
- DevNpsh8 = deviation DevNpsh between +2 and +2.5 [m]
- DevNpsh9 = deviation DevNpsh between +2.5 and +3 [m]
- DevNpsh10 = deviation DevNpsh > + 3 [m]

An additional counter is used to determine downtime (DevNpsh0).

In contrast to the flow rate histogram, this histogram is divided into two bar charts. The counters 0 to 9 (DevNpsh0...DevNpsh9) are shown in the chart on the left, while the tenth counter (DevNpsh10) is shown in the chart on the right. The chart is divided into two because, under normal circumstances, the deviation from the NPSH characteristic at 99% is in the range "> 3m". The other counters would then be too small to appear on the chart. For this reason, the maximum value of the Y axis on the chart on the left is derived from the difference of the tenth counter (DevNpsh10) at 100%.

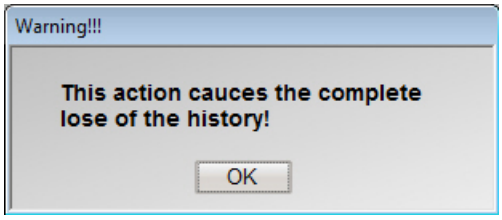
When DevNpsh10 = 99%, therefore, the Y axis on the left is shown with a maximum value of 1% and the values of the other counters, which together comprise 1% of the total recording time, can be displayed and analyzed.

The Y axis scaling is dynamic:

- **Y0-9:** `0 -> (100% - DevNpsh10) / 2 -> 100% - DevNpsh10`
- **Y10:** `0 -> 50 -> 100%`

For more information, refer:Bar graph (Page 376)

**(3) Display of the characteristic values**
The "Recording duration" field displays the time period in (`LoadTime`) over which these statistics were collected. The reset button can be used to reset all the stored values to zero and restart. The following message is displayed prior to this:

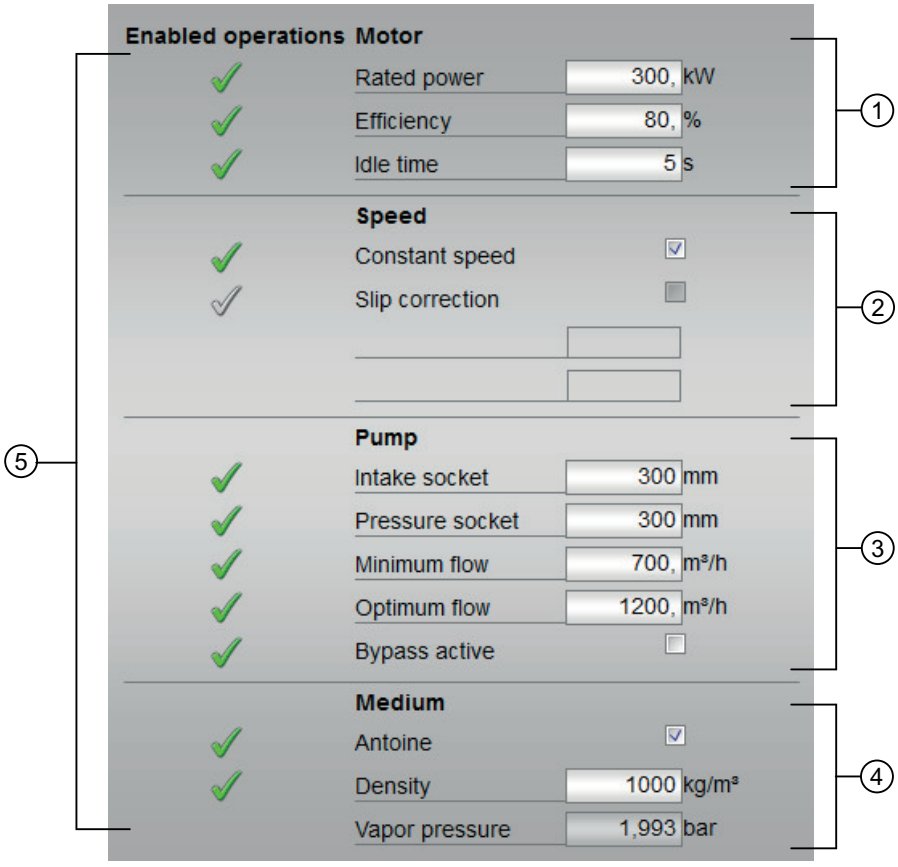



All collected statistics will be lost with this action!

**Note**

When the upper limit of the flow rate range (FlowMax) is changed, the counters are also reset.

### 4.1.10.6 PumpMon parameter view




**(1) Motor parameters**

You can change the following parameters here:

- "Rated power": Rated power (`PoRate`)
- "Efficiency": Efficiency (`Eta`)
- "Idle time": Message suppression running period (`IdleTime`)

**Note**

If one of the parameters for "Converter active" (`ConvAct`) or "Characteristic active" (`PolyAct`) has been set on the block (ES), the field to enter the efficiency is not visible.

**(2) Speed parameters**

The following functions may be enabled here:

- "Constant speed": Motor with variable or constant speed (`ConstSpd`)
- "Slip correction": Slip correction (`SlipCorr`)

You can also change the following parameter:

- "Rated speed": Rated speed (`SpdRate`)

The following parameter is only displayed and cannot be changed:

- "Speed": Actual speed (`siSpd.Value`)

**Note**

If the pump is a constant speed pump, the rated speed, the actual speed, and slip correction are not needed. In this case, the control permission for the slip correction checkbox is revoked and the rated and actual speeds are hidden.

**(3) Pump parameters**
You can change the following parameters here:

- "Intake socket": Diameter of the intake stub (`InSocket`)
- "Pressure socket": Diameter of the discharge stub (`PresSocket`)
- "Minimum flow": Minimum flow rate (`FlowMin`)
- "Optimum flow": Optimum flow rate (`FlowOpt`)

The following functions can also be enabled:

- "Bypass active": Enable bypass calculation (`Bypass`)

**(4) Medium parameters**
For a wide range of different media, the vapor pressure (`siP_Vapor`) can be calculated as a function of the temperature in accordance with the Antoine equations. The parameters for water at 1-100°C are set as default.

The delivery height (`Density`) of the medium is also required to calculate the delivery height from the measured pressure values.

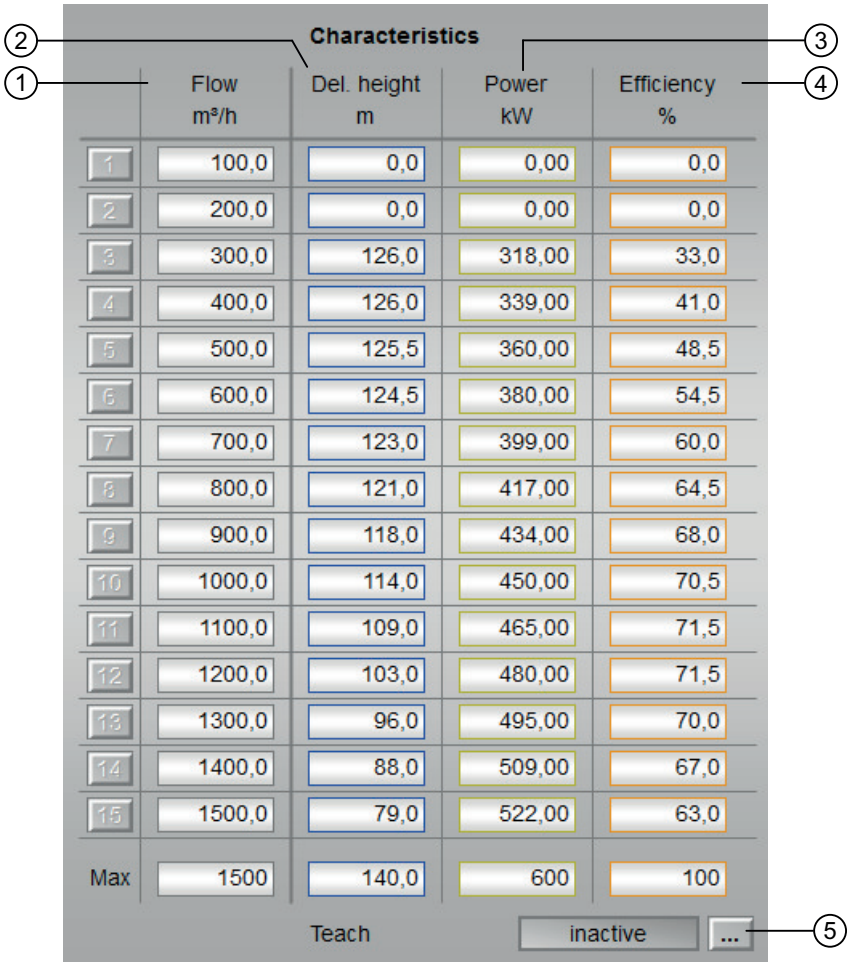
If this equation is not to be used, it can be disabled by unchecking the checkbox (`Antoine`). The vapor pressure is then supplied via the block input `P_Vapor`. The input can be interconnected and so cannot be used via the faceplate.

**(5) Enabled operations**
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark:** The OS operator can control this parameter
- **Gray check mark:** Access to this parameter is temporarily disabled for the OS operator due to process conditions
- **Red cross:** The OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

### 4.1.10.7 PumpMon characteristic parameter view 1




The support values for the characteristic must either be taken from the data sheet for the pump and entered here, or taught by means of the teach function (see below). The former scenario provides a direct comparison with the theoretical optimum pump behavior. The latter scenario, however, can also be used to record the "ideal status" under non-ideal circumstances (e.g. pressure measurement not taken directly at the pump outlet, etc.) to obtain a reference for subsequent deviations.

**(1) Flow rate parameters**
The 15 flow rate support values each specify the x axis for the delivery height, power, and efficiency characteristics.

The display is within the range:

```
0 <= Flow1 <= ... <= Flow15 <= FlowMax
```

Support values at the beginning or end of the characteristic with the value "zero" are not shown.

The values for the flow rate always have the unit [m³/h].

**Note**

When `FlowMax` is changed, the values saved in the histograms are lost. The same warning message is shown as when resetting the histograms.

**(2) Delivery height parameters**
The parameters `DelHi1` to `Delhi15` for the delivery height characteristic can be entered in this area. Parameters that have the value 0 (see parameter 1) are ignored in the characteristic curve.

The values for the delivery height always have the unit [m].

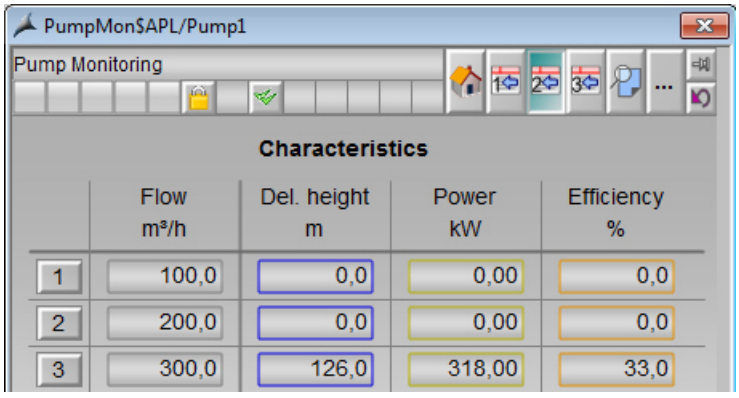
**(3) Power parameters**
The parameters `Power1` to `Power15` for the power characteristic are entered in this area. Parameters that have the value 0 are ignored on the characteristic curve.

The values for the power always have the unit [kW].

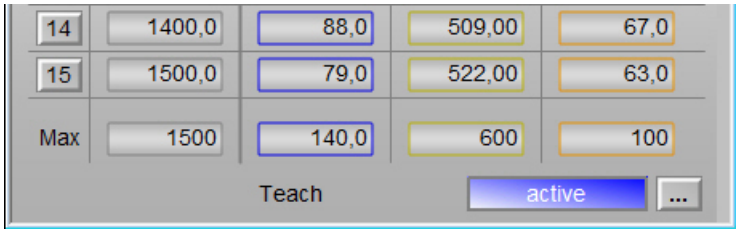
**(4) Efficiency parameters**
The parameters `Eta1` to `Eta15` for the efficiency characteristic are entered in this area. Parameters that have the value 0 are ignored on the characteristic curve.

The values for the efficiency always have the unit [%].

**(5) Teach function**
The teach function is used to apply the characteristic values calculated from the current process values for an operating point as support values. The operating point must be approached for this purpose and it is necessary to wait until the values are steady.

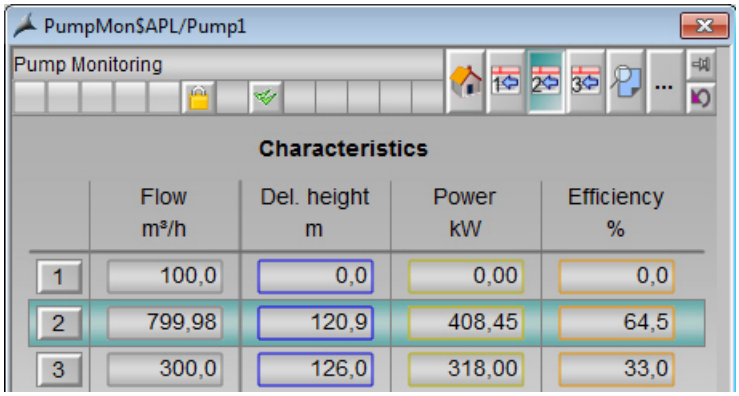



...

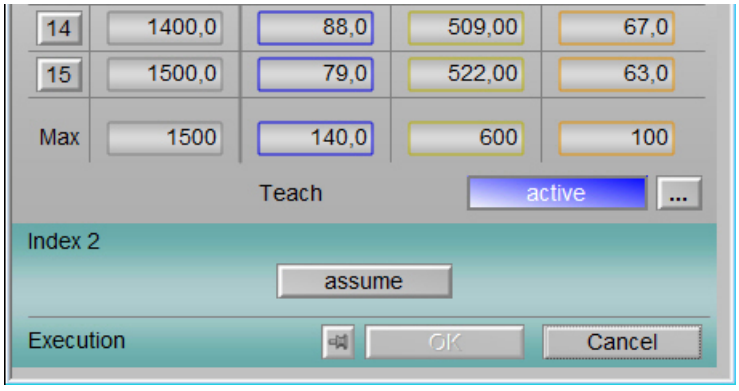



After enabling the "Teach" button, the operator control permissions for all characteristic parameters are revoked and the numbered buttons at the start of the parameter line are enabled.

Clicking on one of these buttons extends the "SelectionBorder" used to display the currently operated element to the entire line. Otherwise, the save parameters are no longer shown – only the parameters currently calculated by the block. The visible values are applied via binary acknowledgment.



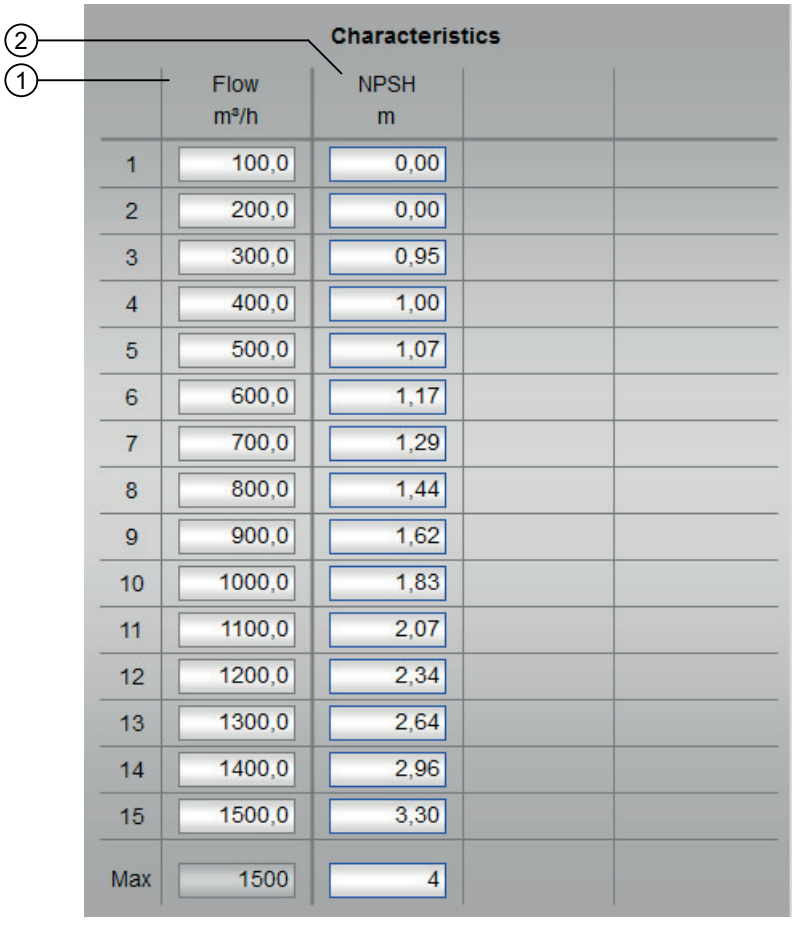

...




**Note**

There is no undo function. Incorrect entires must be corrected individually and "by hand".

### 4.1.10.8 PumpMon characteristic parameter view 2




The NPSH characteristic cannot be configured by means of the teach function, which is why it is assigned separate support values for the flow rate.

**(1) Flow rate parameters**
The 15 support values for the flow rate define the x axis for the NPSH characteristic.

The display is within the range:

```
0 <= FlowNp1 <= ... <= FlowNp15 <= FlowMax
```

Support values at the beginning or end of the characteristic with the value "zero" are not shown. The values for the flow rate always have the unit [m³/h].

**Note**

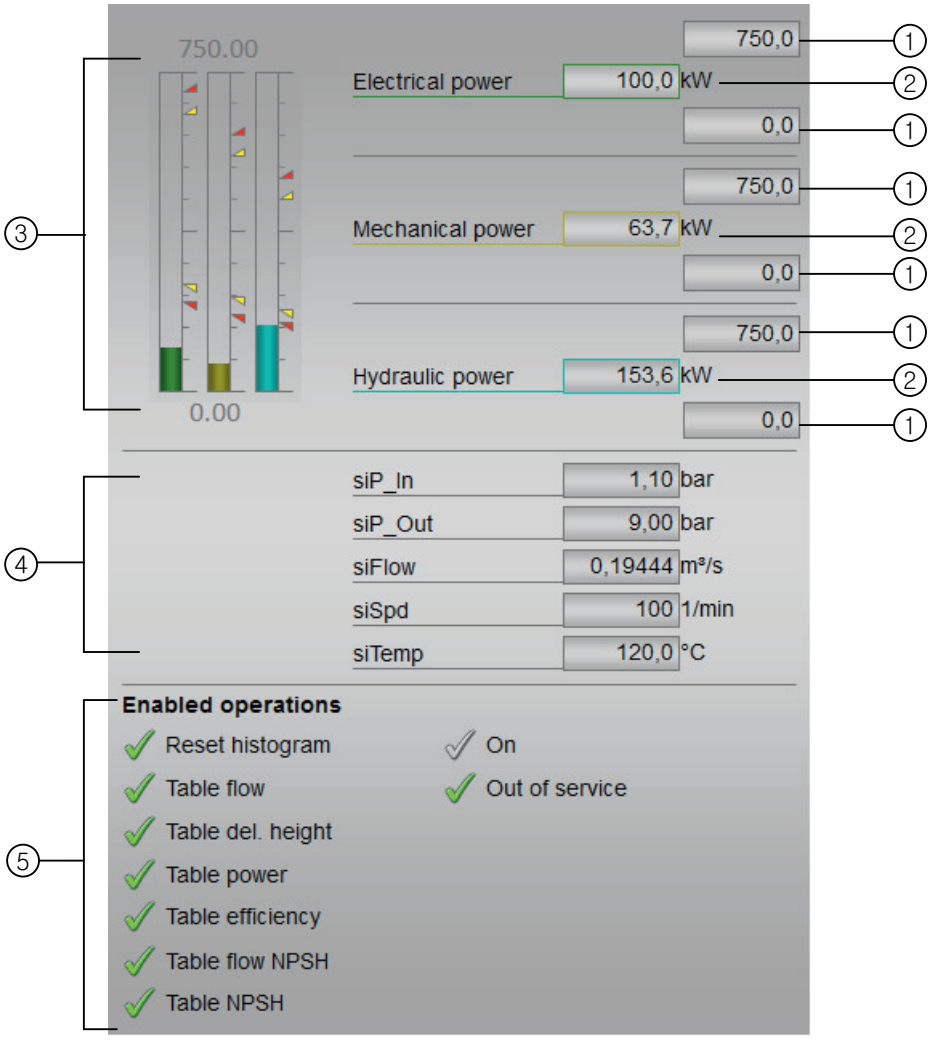
The value for the maximum flow rate (`FlowMax`) is only displayed in this view. This is the same value as the value for the other three characteristics from the "Characteristic parameter view 1 of PumpMon" view. Any change to the value must be made in that view.

**(2) NPSH parameters**

The parameters `Npsh1` to `Npsh15` for the NPSH characteristic and the maximum value `NpshMax` are entered in this area. Parameters that have the value 0 are ignored in the characteristic curve.

The NPSH values always have the unit [m].

### 4.1.10.9 Preview of PumpMon




**Note**

The "Preview" label has been chosen for reasons of compatibility with other blocks of the Advanced Process Library. The displayed process values are the **current** process values. The "Preview" refers to the enabled operations.

**(1) Upper and lower scale range of the power values**
These values provide information about the display range for the bar graph of the power values. The scale range is specified in the engineering system (`PowerOpScale`).

**(2) Display of the power values including signal status**
This area shows you the current power values with the associated signal status:

- Electrical power (`siPoElec.Value`)
- Mechanical power (`siPoMech.Value`)
- Hydraulic power (`siPoHydr.Value`)

**(3) Bar graph for the power values**
This area shows you the current power values in form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

**(4) Normalized process values including signal status**
The normalized process values with the associated signal status are shown in this area:

- Input pressure (`siP_In.Value`)
- Output pressure (`siP_Out.Value`)
- Flow rate (`siFlow.Value`)
- Speed (`siSpd.Value`)
- Temperature (`siTemp.Value`)

**(5) Enabled operations**
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark:** The OS operator can control this parameter
- **Gray check mark:** Access to this parameter is temporarily disabled for the OS operator due to process conditions
- **Red cross:** The OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

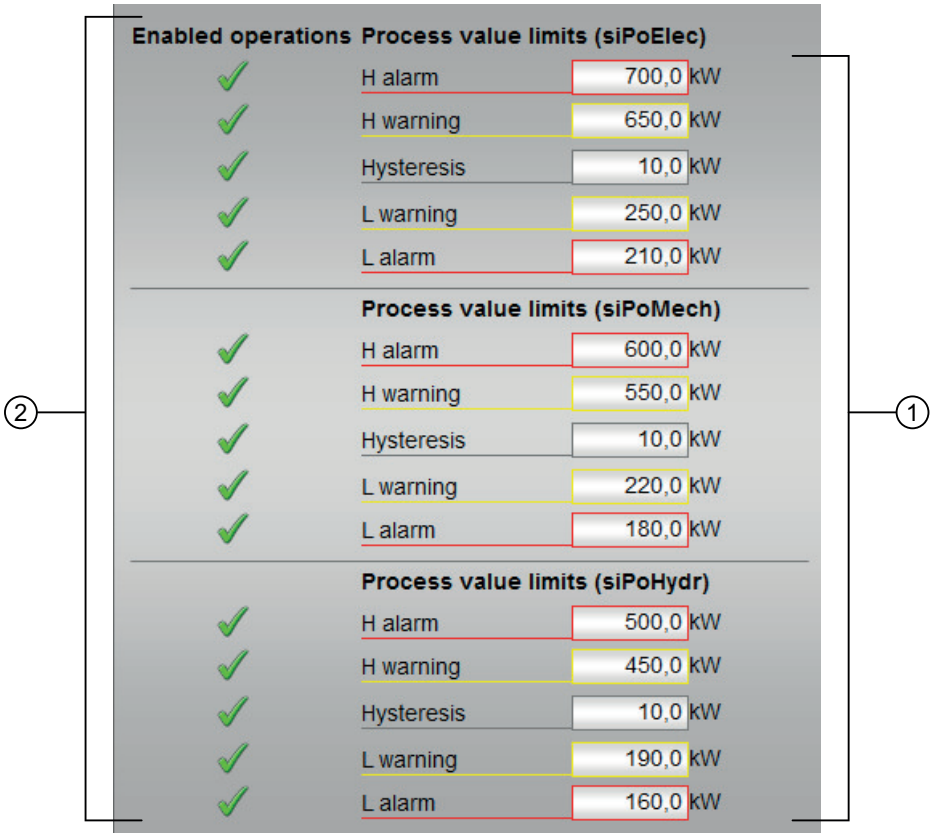
The following enabled operations are shown here:

- "Reset histogram": You can reset the block histograms.
- "Flow rate characteristic": You can change the flow rate characteristic values.
- "Delivery height characteristic": You can change the delivery height characteristic.
- "Power characteristic": You can change the power characteristic.
- "Efficiency characteristic": You can change the efficiency characteristic.
- "NPSH flow rate characteristic": You can change the flow rate characteristic values for the NPSH characteristic.
- "NPSH characteristic": You can change the NPSH characteristic.
- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.

### 4.1.10.10 PumpMon limit value view

Process value limits can be set in this view:

- Electrical power
- Mechanical power
- Hydraulic power




**(1) Process value limits**
You can enter the limits for the process values in this area.

You can change the following limits:

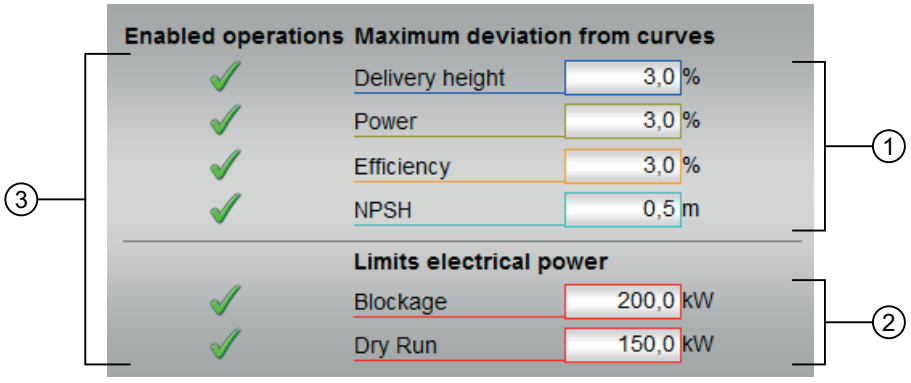
- "H alarm": High alarm
- "H warning": High warning
- "Hysteresis"
- "L warning": Low warning
- "L alarm": Low alarm

**(2) Enabled operations**
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark:** The OS operator can control this parameter
- **Gray check mark:** Access to this parameter is temporarily disabled for the OS operator due to process conditions
- **Red cross:** The OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

### 4.1.10.11 PumpMon tolerances view




**(1) Maximum deviation from the characteristic curves**
In this area, you can enter the tolerances for the deviation from the characteristic curves.

You can change the following tolerances:

- "Delivery height": Maximum deviation from the delivery height characteristic (`DelHiTol`)
- "Power": Maximum deviation from the power characteristic (`PowerTol`)
- "Efficiency": Maximum deviation from the efficiency characteristic (`EtaTol`)
- "NPSH": Maximum deviation from the NPSH characteristic (`NpshTol`)

**(2) Limits for the electrical power**
The following limits can be entered in this area:

- "Blockage": Electrical power in the event of blockage (`PoBlk`)
- "Dry running": Electrical power in the event of dry running (`PoDryRun`)

**Note**

The entries apply to operation under rated conditions. With variable-speed drives, the limits are changed in accordance with the speed.

**(3) Enabled operations**
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

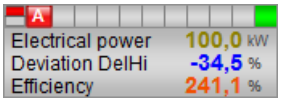
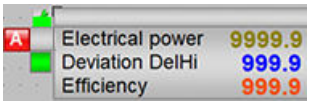
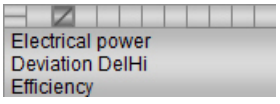
- **Green check mark:** The OS operator can control this parameter
- **Gray check mark:** Access to this parameter is temporarily disabled for the OS operator due to process conditions
- **Red cross:** The OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

### 4.1.10.12 Block icon for PumpMon

A block icon with the following functions is available to you:

- Process tag type
- Violations of alarm, warning, and tolerance limits
- Operating modes
- Signal status, release for maintenance
- Memo display
- Operating status of the pump
- Mechanical power
- Relative distance from the delivery height characteristic
- Efficiency

The block icon from template @TemplateAPLV9.PDL:

| Symbols | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in full display |
|  | 2 | Block icon in full display |
|  | - | Block icon in the "Out of service" operating mode |

You can find additional information on the block icon and the operator control options in the block icon in the following sections:

- "Configuring the block icons (Page 267)"
- "Block icon structure (Page 260)"
- "Operation via the block icon (Page 269)".

# 4.2 PrDrpMon - Pressure drop monitoring

## 4.2.1 Description of PrDrpMon

### Object name (type + number) and family

Type + number: FB 1952

Family: CondMon

Number of process objects: 10 PO

### Area of application of PrDrpMon

The objective is status monitoring of plant components with the `PrDrpMon` function block within the framework of SIMATIC PCS 7 plant asset management. The monitoring concerns pressure loss of (or pressure drop to) plant components such as filters, separators, heat exchangers, pipes, etc. depending on flow, that is, monitoring of flow resistance. This type of monitoring can be useful for all plant components whose flow resistance changes in an undesirable manner during operation due to deposits, blockages, etc. This low-cost solution is based on evaluation of sensor signals that are already present in the control system and contrasts with expensive condition monitoring systems that rely on special additional sensors, such as acoustic emission or acceleration sensors.

### Mode of operation

Any limit violations in the nominal device operating range and deviations from the expected characteristic curve are visualized to the operator and made available via the block outputs for further processing. All the values can be processed further with the usual PCS 7 tools (calculations, archiving, curve recording, alarm history, etc.).

The block itself has a purely diagnostic function and, as such, does not intervene directly in the operation of the device. This means that it can be used (including as a retrofitted device) without having to worry about affecting the process. If required, active intervention (e.g., closing the input valve if there is a risk of a blockage) can be undertaken by evaluating the block outputs.

The following views are provided:

- Differential pressure characteristic curve: Display of the normalized differential pressure as a function of flow and current viscosity range with current operating point of the device.
- Various views for (one-time) input of device and medium parameters

### Configuration

Install the block in a cyclic interrupt OB (OB30 to OB38) in the CFC Editor. The block is also installed automatically in the startup OB (OB100).

### Dynamic input values

The block requires the current viscosity, the flow, and the differential pressure across the device as input data. In addition, a binary signal from a SteadyState block is needed in each case for determining the stationary state of the respective value. These signals must be provided by upstream blocks.

The following input signals must be interconnected:

- Flow of pumped media (Flow)
- Binary signal for stationary state of the flow (FlowStat)
- Input pressure (P_In)
- Binary signal for stationary state of the input pressure (P_InStat)
- Output pressure (P_Out)
- Binary signal for stationary state of the output pressure (P_OutStat)
- Viscosity (Visc)
- Stationary state of the viscosity (ViscStat)
  Instead, the viscosity can be defined as a constant value using Feature Bit 11.

### Static input values

The following data must be known and entered as static input data (most parameters can be entered via the faceplate):

- Minimum and maximum value for viscosity
- Maximum value for flow and differential pressure
- Differential pressure setpoints of different flow values for the various viscosity ranges

### Startup characteristics

Use Feature Bit 0 (Setting the startup characteristics) to define the startup characteristics for this block.

- Bit = 0: The most recently stored values are reset at startup.
- Bit = 1: The most recently used values are sent to the following output parameters at startup:
  - ActFlow1 to ActFlow15
  - ActDeltaP1 to ActDeltaP15
  - DevDeltaP
  - RelDeltaP
  - siDeltaP
  - siFlow
  - siP_In
  - siP_Out

**Note**

Additional information about `Feature.Bit0` can be found in "Set startup characteristics (Page 148)"

### Status word assignment for the `Status1` parameter

For a description of the individual parameters, refer to the section "I/Os of PrDrpMon (Page 550)".

| Status bit | Parameter |
|---|---|
| 0 - 2 | Not used |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `OnAct.Value` |
| 6 – 10 | Not used |
| 11 | `DeltaP_Act.Value` |
| 12 | `CritDeltaP_Act.Value` |
| 13 - 27 | Not used |
| 28 | Internal error |
| 29 - 31 | Not used |

### Status word assignment for `Status2` parameter

For a description of the individual parameters, refer to the section "I/Os of PumpMon (Page 505)".

| Status bit | Parameter |
|---|---|
| 0 | Not used |
| 1 | `Feature.Bit10` |
| 2 | `Feature.Bit11` |
| 3 | `ViscMin <= Visc.Value <= ViscMax` |
| 4 | `Visc.Value < ViscMin` |
| 5 | `Visc.Value > ViscMax` |
| 6 - 31 | Not used |

### Status word assignment for `Status3` parameter

For a description of the individual parameters, refer to the section "I/Os of PumpMon (Page 505)".

| Status bit | Parameter |
|---|---|
| 0 | Not used |
| 1 | `ViscIndex = 1` |
| 2 | `ViscIndex = 2` |
| 3 | `ViscIndex = 3` |

| Status bit | Parameter |
|---|---|
| 4 | ViscIndex = 4 |
| 5 | ViscIndex = 5 |
| 6 | Visc.Value in Range 1 |
| 7 | Visc.Value in Range 2 |
| 8 | Visc.Value in Range 3 |
| 9 | Visc.Value in Range 4 |
| 10 | Visc.Value in Range 5 |
| 11 | RelDeltaP > DeltaP_Tol |
| 12 | RelDeltaP > CriticalDeltaP |
| 13 - 31 | Not used |

## 4.2.2 PrDrpMon modes

The block can be operated using the following modes:

- On
- Out of service

### "On"

General information about the "On" operating mode can be found in the section "On (Page 77)".

### "Out of service"

General information about the "Out of service" operating mode can be found in the section "Out of service (Page 77)".

## 4.2.3 PrDrpMon mode of operation

The PrDrpMon block is supplied with current process values in the CFC via the inputs Flow (flow), P_In (input pressure), P_Out (output pressure), and Visc (viscosity of the flow medium).

The binary inputs FlowStat, P_InStat, P_OutStat, and ViscStat are used to signal whether a new stationary operating point has been reached (in the case of non-constant viscosity).

### Preprocessing

The process values are normalized to SI units via the *Offs and *Fact inputs. The normalized values are output again at the output. The status signals of the process values (quality code) are not evaluated but are instead output again at the output.

- `siFlow.Value = FlowOffs + Flowfact * Flow.Value`
  (normalize to $m^3/h$)
- `siP_In.Value = P_InOffs + P_InFact * P_In.Value`
  (normalize to bar)
- `siP_Out.Value = P_OutOffs + P_OutFact * P_Out.Value`
  (normalize to bar)

### Stationary processes

All 4 process values have a separate input for information on the stationary state of the respective process value:

| | |
|---|---|
| `FlowStat` | Stationary state of the flow |
| `P_InStat` | Stationary state of the input pressure |
| `P_OutStat` | Stationarity of the output pressure |
| `ViscStat` | Stationary state of the viscosity |

The inputs are interconnected with the `StatAct` output of the corresponding SteadyState block.

The stationary state signals for the input and output pressure are ANDed internally and then evaluated.

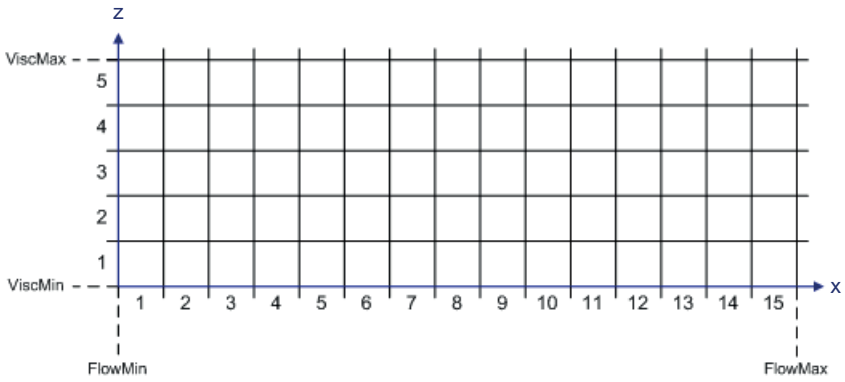
### Differential pressure characteristic field

In contrast to existing condition monitoring blocks that can display various two-dimensional characteristic fields, the `PrDrpMon`block must additionally consider a third dimension (viscosity) because the differential pressure characteristic curve is strongly dependent on it.

If the viscosity is constant enough in an application, this third dimension can also be suppressed again. In this case, `Feature.Bit11 = TRUE` must be assigned. Then only flow and differential pressure are evaluated for the characteristic field analysis.

**Note**

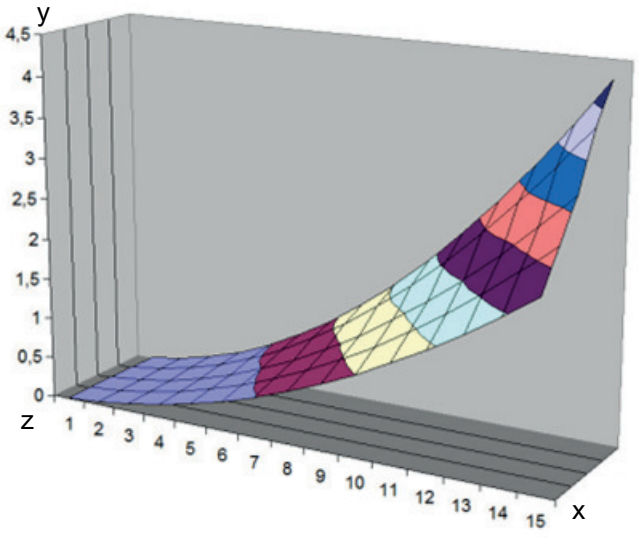
If the viscosity is not available as a measured value in an application, the media temperature can be interconnected by way of substitute. This changes the direction of dependency (high temperatures signify low viscosity) but this is irrelevant for the functioning of the block. For this reason, units for viscosity are also consistently omitted in the block. All labels and parameter name always refer to viscosity, however




**z** = Viscosity

**x** = Flow rate

Like for `PumpMon`, the X-dimension (Flow) is divided into 15 equal-sized areas (0 to `FlowMax`). The Y-dimension represents the pressure difference between the output pressure (`P_Out`) and input pressure (`P_In`) (0 to `DeltaP_Max`), and the Z-dimension (Visc) is divided into 5 equal-sized areas (`ViscMin` to `ViscMax`) for the characteristic field analysis. If the current measured value Visc is outside the range `ViscMin` to `ViscMax`, no points are taught or displayed. Instead, the text "`Visc < ViscMin`" or "`Visc > ViscMax`" is displayed in the standard view of the faceplate.




**x** = Flow rate ($m^3/h$)

**y** = Pressure drop (bar)

**z** = Viscosity ($m^2*s^{-1}$)

## Set differential pressure characteristic curve

For the sake of clarity, there is no three-dimensional display of the characteristic field in the faceplate. Only the sectional view belonging to the measured viscosity is calculated by the characteristic field and displayed as a two-dimensional characteristic curve.

This yields a separate two-dimensional set characteristic curve for each of the five viscosity ranges.

The interpolation points of the set differential pressure characteristic curve are written to the input parameters `Flow1`...`Flow15` and `DeltaP1`...`DeltaP15` for the display in the faceplate, depending on the currently selected viscosity range (ViscIndex).

When the viscosity range is changed (`ViscIndex`), the values of the individual interpolation points must first be saved in a static array before the static values of the new viscosity range are copied to the input parameters.

**Note**

If constant viscosity is assumed (`Feature.Bit11 = TRUE`), the `ViscStat` input is not evaluated. Only the `Flow1`...`Flow15` and `DeltaP1`...`DeltaP15` input parameters are considered.

### Automatic acquisition of the actual differential pressure characteristic curve

The actual differential pressure characteristic curve is acquired automatically depending on `Feature.Bit10`. When `Feature.Bit10 = TRUE`, a new value is acquired when all input signals are "stationary" (`FlowStat = 1`, `P_InStat = 1`, `P_OutStat = 1`, `ViscStat = 1`).

Another value is only saved when one of the four signals was "not stationary" in the meantime (e.g., `FlowStat = 0`).

If a new measured value has been saved, the interpolation points that do not yet have a measured value will be recalculated by interpolation.

**Note**

Creeping signal changes must be detected by a corresponding setting of SteadyState.

**Note**

If constant viscosity is assumed (`Feature.Bit11 = TRUE`), the `ViscStat` input is not evaluated.

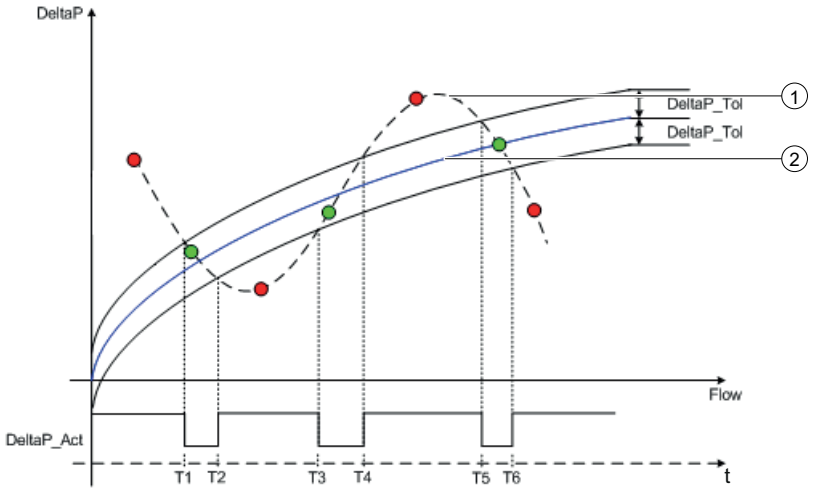
## 4.2.4 PrDrpMon functions

The functions for this block are listed below.

### Tolerance monitoring of the characteristic curves

Because the characteristic field is only expanded by individual interpolation points, an interpolation with the characteristic field is required for each comparison of the operating point, as shown in Figure 3.

For this, the four measured values around the operating point are considered in each case. An auxiliary point is calculated in each case between the two pairs that lie in the same viscosity range. These two auxiliary points are determined in such a way that they have the same flow value as the operating point. Now, it is only necessary to interpolate between the two auxiliary points in order to calculate the setpoint.




(1) Movement of the operating point over time

(2) Differential pressure characteristic

t = time

The deviation of the operating point from the two-dimensional set differential pressure characteristic curve calculated in this way is output in outputs `RelDeltaP` (relative in [%]) and `DevDeltaP` (absolute in [bar]). If the operating point exits the specified tolerance band (`DeltaP_Tol`), the `DeltaP_Act` output is set and the operating point changes from green to red in the faceplate.

The user assigns the critical relative deviation to the `CriticDeltaP` input parameter. When this parameter is reached, the functional capability of the monitored component is endangered and an immediate maintenance measure is required. In this case, the `CritDeltaP_Act` output is set to 1.

For flow values less than `MonFlowMin`, the differential pressure is not monitored. The value can be specified as a relative or static value ().

## Wear reserve

Depending on the relative deviation `RelDeltaP`, the wear reserve is calculated and output only at the `WearResDeltaP` output. The wear reserve is calculated as follows:

$$WearResDeltaP = 1 - \left( \frac{RelDeltaP}{CriticDeltaP} \right)$$

### Forming the signal status for blocks

This block has the standard function "Forming and outputting the signal status for technological blocks (Page 117)".

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `Flow.ST`
- `FlowStat.ST`
- `P_In.ST`
- `P_InStat.ST`
- `P_Out.ST`
- `P_OutStat.ST`
- `Visc.ST`
- `ViscStat.ST` (if `Feature.Bit11 = FALSE`)

The following dependencies apply to the formation of the output-side signal status:

| Signal status | Dependencies |
|---|---|
| siFlow.ST | Flow.ST |
| siP_In.ST | P_In.ST |
| siP_Out.ST | P_Out.ST |

### Selecting a unit of measurement

This block provides the standard function "Selecting a unit of measure (Page 240)".

### Configurable reactions using the `Feature`parameter

An overview of all the reactions that are provided by the `Feature` parameter is available in section "Configurable response using the `Feature` I/O (Page 139)". The following modes of behavior are available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Setting the startup characteristics |
| 1 | Characteristics for the out of service mode |
| 6 | Minimum flow for monitoring the operating point is absolute/relative |
| 10 | Automatic acquisition of the actual differential pressure characteristic curve |
| 11 | Viscosity is constant |
| 26 | Behavior of the switching points in "Out of service" mode |
| 28 | Disabling of switching points |
| 29 | Signaling limit violation |

### Operator control permissions

This block provides the standard function "Operator control permissions (Page 285)".

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 – 9 | Not used |
| 10 | 1 = Operator can apply the actual differential pressure characteristic curve as the set characteristic curve |
| 11 | 1 = Operator can reset the actual differential pressure characteristic curve. |
| 12 | 1 = Operator can change the value for the on delay |
| 13 - 31 | Not used |

The block has the following permissions for the `OS1Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can change the maximum flow value |
| 1 | 1 = Operator can change the minimum flow value |
| 2 | 1 = Operator can set the viscosity range in the parameter view to 1. |
| 3 | 1 = Operator can set the viscosity range in the parameter view to 2. |
| 4 | 1 = Operator can set the viscosity range in the parameter view to 3. |
| 5 | 1 = Operator can set the viscosity range in the parameter view to 4. |
| 6 | 1 = Operator can set the viscosity range in the parameter view to 5. |
| 7 - 14 | Not used |
| 15 | 1 = Operator can change the limit for the deviation of the differential pressure characteristic curve |
| 16 | 1 = Operator can change the critical limit for the deviation of the differential pressure characteristic curve |
| 17 | 1 = Operator can change the hysteresis for the limit for the deviation of the differential pressure characteristic curve |
| 18 - 31 | Not used |

## SIMATIC BATCH functionality

This block does not support "SIMATIC BATCH functionality".

## 4.2.5 PrDrpMon messaging

### Message behavior

The PrDrpMon block does not generate messages or maintenance alarms. It simply provides several binary message outputs that are set when the permissible deviation is exceeded. Messages may be generated for specific applications by a separate event block, and maintenance alarms by a downstream AssetMon block (included in the standard package of PCS 7). Events that are relevant for the plant operator can also be reported via interconnection of PrDrpMon message outputs to free message inputs of the associated technological block.

## 4.2.6 PrDrpMon I/Os

### I/Os of PrDrpMon

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AdoptActDeltaP_Op* | 1 = Adopt the current actual differential pressure characteristic curve | BOOL | FALSE |
| BatchEn | Enable allocation | BOOL | FALSE |
| BatchID | Batch ID | DWORD | 16#0 |
| BatchName | Batch name | SRING[32] | "" |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| CriticalDeltaP | Critical relative deviation of the set differential pressure characteristic curve [%] | REAL | 50.0 |
| DeltaP1 | y value 1, differential pressure [bar] | REAL | 0.1 |
| DeltaP2 | y value 2, differential pressure [bar] | REAL | 0.4 |
| DeltaP3 | y value 3, differential pressure [bar] | REAL | 0.9 |
| DeltaP4 | y value 4, differential pressure [bar] | REAL | 1.6 |
| DeltaP5 | y value 5, differential pressure [bar] | REAL | 2.5 |
| DeltaP6 | y value 6, differential pressure [bar] | REAL | 3.6 |
| DeltaP7 | y value 7, differential pressure [bar] | REAL | 4.9 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| DeltaP8 | y value 8, differential pressure [bar] | REAL | 6.4 |
| DeltaP9 | y value 9, differential pressure [bar] | REAL | 8.1 |
| DeltaP10 | y value 10, differential pressure [bar] | REAL | 10.0 |
| DeltaP11 | y value 11, differential pressure [bar] | REAL | 12.1 |
| DeltaP12 | y value 12, differential pressure [bar] | REAL | 14.4 |
| DeltaP13 | y value 13, differential pressure [bar] | REAL | 16.9 |
| DeltaP14 | y value 14, differential pressure [bar] | REAL | 19.6 |
| DeltaP15 | y value 15, differential pressure [bar] | REAL | 22.5 |
| DeltaP_Hyst | Hysteresis for the limit monitoring of the differential pressure | REAL | 5.0 |
| DeltaP_Max | Maximum differential pressure (indicates the maximum value for the y-axis) [bar] | REAL | 30.0 |
| DeltaP_Tol | Maximum deviation from the set differential pressure characteristic curve [%] | REAL | 10.0 |
| Feature | 1 = Valve open feedback signal | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| Flow | Interconnectable input for the flow | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Flow1 | x value 1, flow [m³/h] | REAL | 2.0 |
| Flow2 | x value 2, flow [m³/h] | REAL | 4.0 |
| Flow3 | x value 3, flow [m³/h] | REAL | 6.0 |
| Flow4 | x value 4, flow [m³/h] | REAL | 8.0 |
| Flow5 | x value 5, flow [m³/h] | REAL | 10.0 |
| Flow6 | x value 6, flow [m³/h] | REAL | 12.0 |
| Flow7 | x value 7, flow [m³/h] | REAL | 14.0 |
| Flow8 | x value 8, flow [m³/h] | REAL | 16.0 |
| Flow9 | x value 9, flow [m³/h] | REAL | 18.0 |
| Flow10 | x value 10, flow [m³/h] | REAL | 20.0 |
| Flow11 | x value 11, flow [m³/h] | REAL | 22.0 |
| Flow12 | x value 12, flow [m³/h] | REAL | 24.0 |
| Flow13 | x value 13, flow [m³/h] | REAL | 26.0 |
| Flow14 | x value 14, flow [m³/h] | REAL | 28.0 |
| Flow15 | x value 15, flow [m³/h] | REAL | 30.0 |
| FlowFact | Flow factor, normalization to m³/h | REAL | 1.0 |
| FlowMin | Minimum flow (indicates the minimum value for the x-axis) [m³/h] | REAL | 0.0 |
| FlowMax | Maximum flow (indicates the maximum value for the x-axis) [m³/h] | REAL | 30.0 |
| FlowOffs | Flow offset, normalization to m³/h | REAL | 0.0 |
| FlowStat | Interconnectable input for the stationary state monitoring of the flow (SteadyState) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| FlowUnit | Unit of measure for flow | INT | 0 |
| MonFlowMin | Flow value from which the flow limits are to be monitored | REAL | 10.0 |
| OnOp | 1 = "On", via OS operator input | BOOL | FALSE |
| OosLi | 1 = "Out of service", via interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator input | BOOL | FALSE |
| OS_Perm | I/O for operator control permissions (see "Functions of PrDrpMon (Page 546)") | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| OS1Perm | I/O for operator control permissions (see "Functions of PrDrpMon (Page 546)") | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| P_In | Interconnectable input for the input pressure | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| P_InFact | Input pressure factor, normalization to bar | REAL | 1.0 |
| P_InOffs | Input pressure offset, normalization to bar | REAL | 0.0 |
| P_InStat | Interconnectable input for the stationary state monitoring of the input pressure (SteadyState) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_InUnit | Unit of measure for the input pressure | INT | 0 |
| P_Out | Interconnectable input for the output pressure | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| P_OutFact | Output pressure factor, normalization to bar | REAL | 1.0 |
| P_OutOffs | Output pressure offset, normalization to bar | REAL | 0.0 |
| P_OutStat | Interconnectable input for the stationary state monitoring of the output pressure (SteadyState) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_OutUnit | Unit of measure for output pressure | INT | 0 |
| RstActDeltaP_Op* | 1 = Reset actual differential pressure characteristic curve | BOOL | FALSE |
| RunUpCyc | Number of cycles during startup (not currently evaluated) | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | 1 = Call of a block stored in this parameter as an additional face-plate in the standard view | ANY | - |
| SelFp2 | 1 = Call of a block stored in this parameter as an additional face-plate in the standard view | ANY | - |
| siFlowUnit | Unit of measure for the normalized flow | INT | 1349 |
| siP_Unit | Unit of measure for the normalized pressure | INT | 1137 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Stepno | Batch step number | DWORD | 0 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |
| Visc | Interconnectable input for the viscosity | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ViscIndex | Index of the currently selected viscosity range for the set differential pressure characteristic curve | INT | 1 |
| ViscMax | Maximum viscosity (indicates the maximum value for the z-axis) [ViscUnit] | REAL | 500.0 |
| ViscMin | Minimum viscosity (indicates the minimum value for the z-axis) [ViscUnit] | REAL | 300.0 |
| ViscStat | Interconnectable input for the stationary state monitoring of the viscosity (SteadyState) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ViscUnit | Unit of measurement for the viscosity | INT | 0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ActFlow1 | x value 1, actual differential pressure characteristic curve [m³/h] | REAL | 2.0 |
| ActFlow2 | x value 2, actual differential pressure characteristic curve [m³/h] | REAL | 4.0 |
| ActFlow3 | x value 3, actual differential pressure characteristic curve [m³/h] | REAL | 6.0 |
| ActFlow4 | x value 4, actual differential pressure characteristic curve [m³/h] | REAL | 8.0 |
| ActFlow5 | x value 5, actual differential pressure characteristic curve [m³/h] | REAL | 10.0 |
| ActFlow6 | x value 6, actual differential pressure characteristic curve [m³/h] | REAL | 12.0 |
| ActFlow7 | x value 7, actual differential pressure characteristic curve [m³/h] | REAL | 14.0 |
| ActFlow8 | x value 8, actual differential pressure characteristic curve [m³/h] | REAL | 16.0 |
| ActFlow9 | x value 9, actual differential pressure characteristic curve [m³/h] | REAL | 18.0 |
| ActFlow10 | x value 10, actual differential pressure characteristic curve [m³/h] | REAL | 20.0 |
| ActFlow11 | x value 11, actual differential pressure characteristic curve [m³/h] | REAL | 22.0 |
| ActFlow12 | x value 12, actual differential pressure characteristic curve [m³/h] | REAL | 24.0 |
| ActFlow13 | x value 13, actual differential pressure characteristic curve [m³/h] | REAL | 26.0 |
| ActFlow14 | x value 14, actual differential pressure characteristic curve [m³/h] | REAL | 28.0 |
| ActFlow15 | x value 15, actual differential pressure characteristic curve [m³/h] | REAL | 30.0 |
| ActDeltaP1 | y value 1, actual differential pressure characteristic curve [bar] | REAL | 0.1 |
| ActDeltaP2 | y value 2, actual differential pressure characteristic curve [bar] | REAL | 0.4 |
| ActDeltaP3 | y value 3, actual differential pressure characteristic curve [bar] | REAL | 0.9 |
| ActDeltaP4 | y value 4, actual differential pressure characteristic curve [bar] | REAL | 1.6 |
| ActDeltaP5 | y value 5, actual differential pressure characteristic curve [bar] | REAL | 2.5 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ActDeltaP6 | y value 6, actual differential pressure characteristic curve [bar] | REAL | 3.6 |
| ActDeltaP7 | y value 7, actual differential pressure characteristic curve [bar] | REAL | 4.9 |
| ActDeltaP8 | y value 8, actual differential pressure characteristic curve [bar] | REAL | 6.4 |
| ActDeltaP9 | y value 9, actual differential pressure characteristic curve [bar] | REAL | 8.1 |
| ActDeltaP10 | y value 10, actual differential pressure characteristic curve [bar] | REAL | 10.0 |
| ActDeltaP11 | y value 11, actual differential pressure characteristic curve [bar] | REAL | 12.1 |
| ActDeltaP12 | y value 12, actual differential pressure characteristic curve [bar] | REAL | 14.4 |
| ActDeltaP13 | y value 13, actual differential pressure characteristic curve [bar] | REAL | 16.9 |
| ActDeltaP14 | y value 14, actual differential pressure characteristic curve [bar] | REAL | 19.6 |
| ActDeltaP15 | y value 15, actual differential pressure characteristic curve [bar] | REAL | 22.5 |
| DevDeltaP | Deviation from the set differential pressure characteristic curve [bar] | REAL | 0.0 |
| ErrorNum | Output of the existing error number. For error numbers that can be output by this block, see PrDrpMon error handling | INT | -1 |
| DeltaP_Act | 1 = Deviation from the set differential pressure characteristic curve is too large | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DeltaPDrift | Indicates the deviation of the actual characteristic curve from the set characteristic curve | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MonFlowMinOut | Absolute flow value from which the differential pressure is monitored | REAL | 0.0 |
| OnAct | 1 = Block is "On" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS1PermLog | Display of OS1Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1Perm | DWORD | 16#FFFFFFFF |
| RelDeltaP | Relative deviation from the set differential pressure characteristic curve [%] | REAL | 0.0 |
| siDeltaP | Indicates the normalized differential pressure [bar] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| siFlow | Indicates the normalized flow [$m^3/h$] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| siP_In | Indicates the normalized input pressure [bar] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| siP_Out | Indicates the normalized output pressure [bar] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Status1 | Status word 1 | DWORD | 16#0 |
| Status2 | Status word 2 | DWORD | 16#0 |
| WearResDeltaP | Wear reserve for set differential pressure characteristic curve | REAL | 0.0 |

### 4.2.7 PrDrpMon block diagram

A block diagram is not provided for this block.

### 4.2.8 Operator control and monitoring

#### 4.2.8.1 Views of PrDrpMon

The PumpMon block has the following views:

- PrDrpMon Standard view (Page 556)
- Parameter view (Page 559)
- Limit view (Page 558)
- Memo view (Page 338)
- Batch view (Page 336)

For general information on the faceplate and block icon, refer to the "Structure of the faceplate (Page 279)" and "Block icon structure (Page 260)" sections.

### 4.2.8.2 PrDrpMon standard view

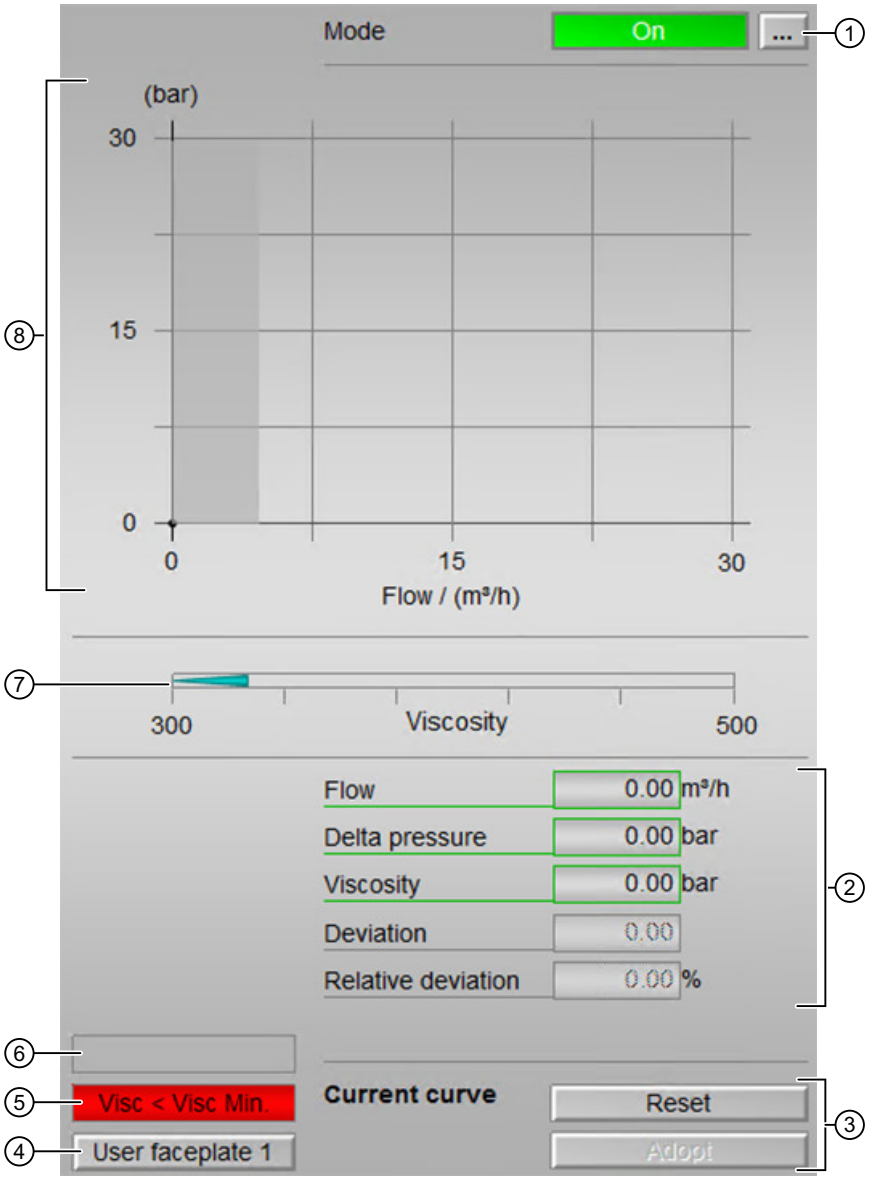



Figure 4-1 Standard view

**(1) Display and switchover of the operating mode**

This area provides information on the operating mode currently in effect. The following operating modes can be shown here:

- On
- Out of service

**(2) Display of the characteristic values**

This area shows the following values:

- Current flow (`siFlow.Value`) (Format: 9999.9)
- Current viscosity (`Visc.Value`) (Format: 999.9)
- Current calculated differential pressure (`siDeltaP.Value`) (Format: 999.9)
- Absolute deviation from the characteristic curve (`DevDeltaP`) (Format: 999.9)
- Relative deviation from the characteristic curve (`RelDeltaP`) (Format: 999.9)

**(3) Functions for the actual characteristic curve**

This area provides a button for adopting and a button for resetting the actual differential pressure characteristic curve.

When the actual characteristic curve is adopted (`AdoptActDeltaP_Op = TRUE`), the actual characteristic curve is written to the corresponding set characteristic curve.

When the actual characteristic curve is reset (`RstActDeltaP_Op = TRUE`), the actual characteristic curve is reset to zero.

**Note**

If `Feature.Bit10 = FALSE`, the buttons for adopting and resetting the actual characteristic curve are disabled.

**(4) Jump button for switching to the standard view of any faceplate**

This display is only visible when the `SelFp1` block input is interconnected.

Use this jump button to reach the standard view of the interconnected block.

**(5) Display area for block states of the characteristic values**

The current viscosity range is displayed here. The corresponding setpoints can be entered for this area in the parameter view.

If the current viscosity range falls outside the limits, "Visc < ViscMin" or "Visc > ViscMax" is displayed accordingly.

**(6) Display when tolerance limit is exceeded**

If the tolerance limit for the differential pressure of the current operating point is exceeded, this is indicated here as text "Diff > DeltaP Tol.". If the critical tolerance limit is exceeded, the text "Diff > Crit. DeltaP Tol." is displayed.

**(7) Display of current viscosity range**

The current viscosity is displayed as a horizontal bar. The scales represent the 5 viscosity ranges.

**(8) Display of actual characteristic curve with current operating point**

The interpolation points for the characteristic curve are entered in the "Characteristic parameters" view.

If the distance of the current operating point exceeds a predefined tolerance, the color of the operating point changes from green to red and a corresponding alarm is triggered (`DeltaP_Act.Value`).

The scaling of the X and Y axis is dynamic

- **X:** 0 -> FlowMax / 2 -> FlowMax
- **Y:** 0 -> DeltaP_Max / 2 -> DeltaP_Max
- The set differential pressure characteristic curve is displayed with the Flow1…Flow15 and DeltaP1…DeltaP15 parameters.
- The actual differential pressure characteristic curve is displayed with the ActFlow1… ActFlow15 and ActDeltaP1…ActDeltaP15 parameters. If Feature.Bit10 = FALSE was assigned, the actual differential pressure characteristic curve is not displayed.
- The unmonitored range (< MonFlowMin) is represented with a semi-transparent gray rectangle.

### 4.2.8.3 PrDrpMon limit view

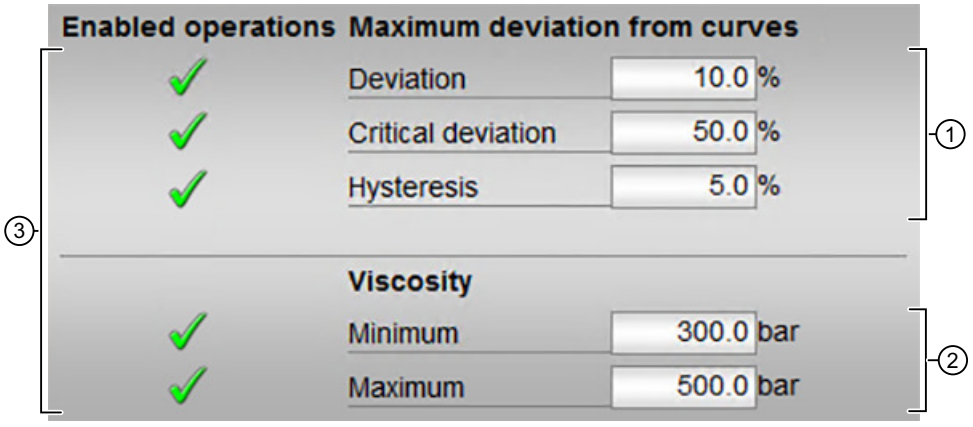



Figure 4-2 Limit value view

**(1) Deviations** from the characteristic curve

In this view, the maximum permitted deviation of the operating point from the set differential pressure characteristic curve DeltaP_Tolis entered (format: 999.9) and the critical relative deviation of the operating point from the set differential pressure characteristic curve CriticDeltaPis entered (format: 999.9) in [%].

Operation requires operator permission "Higher process operations".

**(2) Viscosity parameters**

In this area, the values for the minimum (ViscMin) and maximum (ViscMax) viscosity are specified (unit of measure is ViscUnit); this can also be substituted with a temperature range.

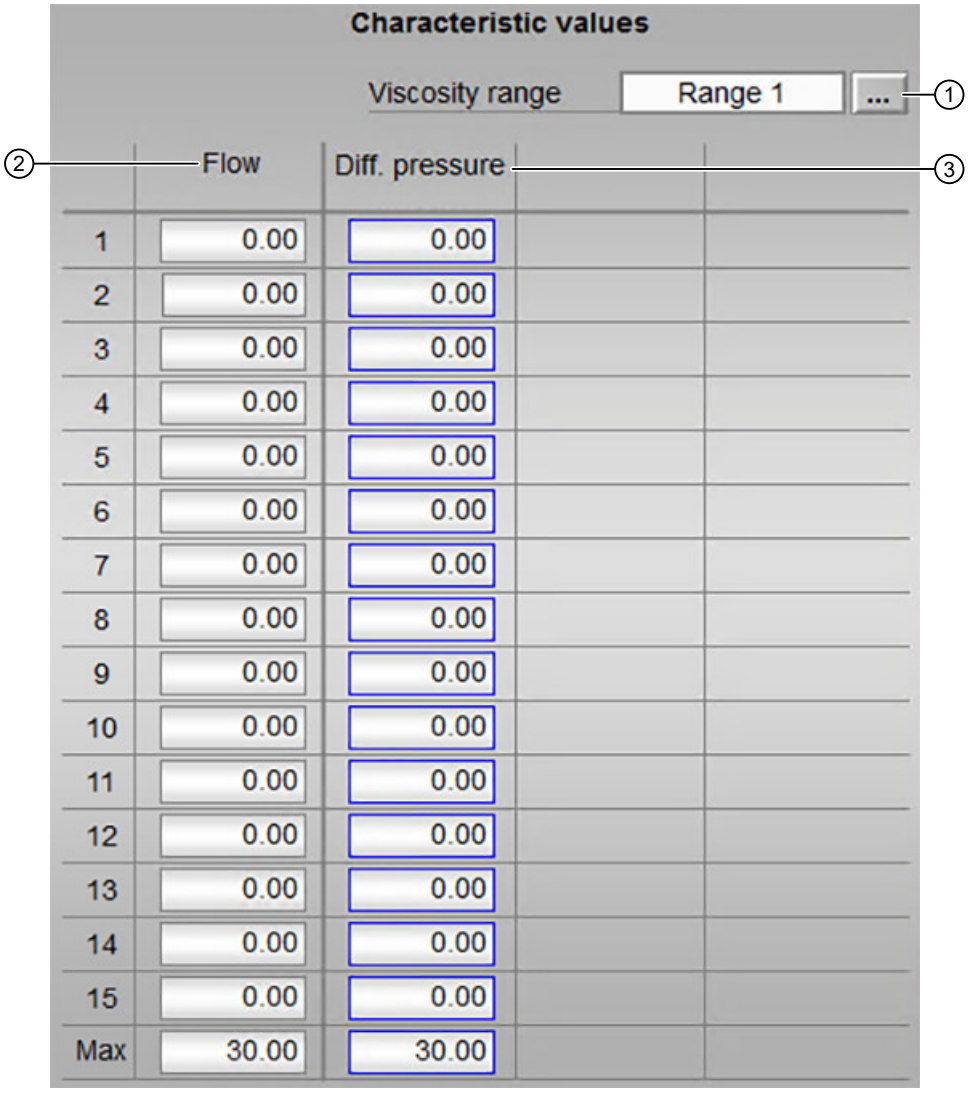
**(3) Enabled operations**

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark:** The OS operator can control this parameter
- **Gray check mark:** Access to this parameter is temporarily disabled for the OS operator due to process conditions
- **Red cross:** The OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

### 4.2.8.4 PrDrpMon parameter view




**(1) Selection of the viscosity range**

The viscosity range to be assigned is selected in this area. The display field of the current viscosity range is shown with a white background.

**(2) Flow parameters**

The 15 interpolation points of the flows (`Flow1` to `Flow15`) indicate the x-axis for the set differential pressure characteristic curve (format: 9999.9).

The display is made within the range 0 <= `Flow1`<= `Flow2`<=, ..., <= `Flow15`<= `FlowMax`.

In addition, the values for the maximum flow (`FlowMax`) can be specified.

**(3) Differential pressure parameters**

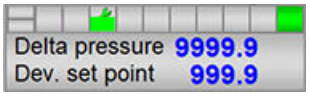
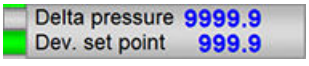
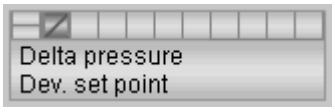
In this area, the differential pressure parameters `DeltaP1`to `DeltaP15`(format: 999.9) and the maximum value `DeltaP_Max`(format: 999.9) are specified. Parameters that have the value 0 are ignored in the characteristic curve.

### 4.2.8.5 Block icon for PrDrpMon

A block icon with the following functions is available to you:

- Process tag type
- Violations of alarm, warning, and tolerance limits
- Operating modes
- Signal status, release for maintenance
- Memo display
- Differential pressure
- Deviation from setpoint

The block icon from template @TemplateAPLV9.PDL:

| Symbols | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in full display |
|  | 2 | Block icon in full display |
|  | - | Block icon in the "Out of service" operating mode |

You can find additional information on the block icon and the operator control options in the block icon in the following sections of the "PCS 7 Advanced Process Library" manual:

- "Configuring the block icons"
- "Block icon structure"
- "Operation via the block icon".

# 4.3 Valve monitoring - VlvMon

## 4.3.1 Description of VlvMon

### Object name (type + number) and family

Type + number: FB 1951

Family: CondMon

Number of process objects: 10 PO

### Area of application

The VlvMon function block offers a low-cost solution for the monitoring and analysis of control valves (continuous action valves). Monitoring is based on the intelligent evaluation of measured values that are often already available in the control system. VlvMon can supplement the self-diagnostics of an electro-pneumatic positioner, for example Sipart PS, or replace it entirely if the positioner has no self-diagnostics or its self-diagnostics is not integrated in the process control system.

Along with pumps, valves are some of the most widely used actuators in process plants. They are subject to various symptoms of wear, such as wear (abrasion, cavitation, corrosion) of the valve plug and/or seat or caking (fouling, deposits) on the valve plug and/or seat. Their behavior during operation can be described by a valve characteristic curve. Valves are mechanical assets which, like pumps, are not registered as field devices in the process control system. However, they are typically pneumatically driven and positioned with an electro-pneumatic positioner. The positioner may be already integrated as intelligent field device in the asset management of the control system.

The status of the valve has a decisive influence on the availability and safety of the entire plant. VlvMon enables status-based monitoring of control valves using various process values. Valve malfunctions or imminent valve failures can thus be detected at an early stage. This improves maintenance planning and increases the availability of the overall plant.

The VlvMon function is used to:

- Warn of valve damage in adverse operating conditions or of wear limits being reached: Continuous operation without valve shutdown, continuous standstill, i.e. long periods of operation without valve motion risking valve caking, exceeding the max. number of strokes, exceeding the max. changes in direction.
- Provide early detection of imminent damage to valve: Deposits or caking, wear, monitoring of reaction times and flow characteristic curves.
- To optimize the valve dimensioning in the long term by means of statistical analysis of the operational data: Frequency distribution of the valve settings.

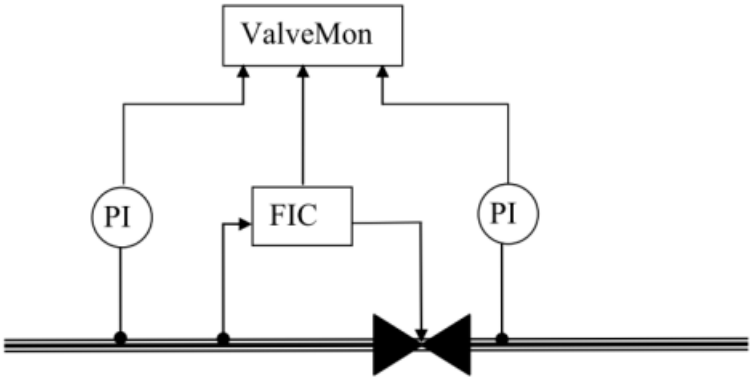
The VlvMon block can be used for each continuous action valve which can be adjusted continuously and returns the current valve position. Pure switching valves (open/closed) are not the subject of consideration. Besides basic status monitoring (monitoring of standstill time, change of direction, etc.), valve motion errors (wear, caking, etc.) are also considered.

If there are additional process values, advanced valve diagnostics can be performed (monitoring of valve characteristic curve based on differential pressure and flow rate, monitoring the supply pressure, etc.).

Relevant process values in the field of valves:

- Flow of the flowing medium
- Inlet pressure (pressure before the valve)
- Outlet pressure (pressure after the valve)
- Valve setpoint
- Valve actual position (feedback about actual valve position)
- Pressure of the compressed air supply

P&ID diagram of a control valve with associated sensors including valve monitoring:




## Mode of operation

When permissible deviations are exceeded, this is signaled to the user by the VlvMon block and made available by block outputs for further processing.

VlvMon does not offer messaging itself, but can be interconnected with the standard PCS 7 message blocks.

The block itself is designed purely for diagnostic purposes and, as such, does not intervene directly in the operation of the control valve. This means that it can be used (including as a retrofitted device) without having to worry about affecting the process. If desired, active intervention can be achieved for specific applications by evaluating the block outputs.

## Configuration

The VlvMon block is dragged from the block catalog and dropped onto the specially prepared CFC. The selection of the insert position and the OB in which the block is installed depends on the insert position of the driver block for the pressure and flow measured values. If the measurement of reaction times of the valve is a priority, both the driver blocks and VlvMon run in a correspondingly rapid cycle (about 10 times faster than the travel times of the valve). The other monitoring functions (characteristic curves, operating hours, etc.), however, are not time-critical. If the reaction times do not have to be monitored in VlvMon, it can be installed in a slow OB to save CPU resources.

The SteadyState block of the Condition Monitoring Library can be integrated to automatically detect the flow characteristic curve. The SteadyState block is used to detect steady states in a dynamic process. It analyzes an input signal and decides without delay whether this signal is steady or not. For more details, refer to the online help of the SteadyState block.

Detection of the flow characteristic curve requires a SteadyState block which must be interconnected as follows:

- Process value flow at the driver block → SteadyState `PV`
- SteadyState `StatAct` → VlvMon `FlowStat`

## Dynamic input values

For control valves with feedback of the valve position, the block requires the following process values:

- Valve setpoint (`MV`)
- Valve actual position (`Rbk`)

For extended monitoring with additional sensors:

- `Flow`
- Pressure before the valve (`P_In`)
- Pressure after the valve (`P_Out`)
- Binary signal for the compressed air supply monitoring (`SuppPres`)

The process signals are interconnected to the associated driver blocks.

## Static input values

The following data must be known and entered as static input data (most parameters can be entered via the faceplate):

- Maximum number of valve strokes (`StrokesMax`)
- Maximum number of changes of direction (`SID_Max`)
- Maximum possible flow through the valve (`FlowMax`)
- Maximum valve reaction time for a complete stroke (`T_RespMax`)
- Maximum allowable idle time without interruption (`T_Idle`)
- Maximum allowable travel time without interruption (`T_Move`)

Flow and reaction time characteristic curves can be determined automatically by the VlvMon block during ongoing plant operation. If this is not possible because no large movements are allowed during ongoing plant operation, the characteristic curves can be taken from manuals and entered manually.

### Startup characteristics

Use `Feature.Bit0` (Setting the startup characteristics) to define the startup characteristics of this block.

- **Bit = 0**: The most recently stored values are reset at startup.
- **Bit = 1:** The most recently used values are output to the following output parameters at startup:
  - `ER`
  - `FlowAct`
  - `FlowDriftAct`
  - `RespUpAct`
  - `RespDnAct`
  - `T_MoveMaxAct`
  - `T_IdleMaxAct`
  - `DeadAct`
  - `StrokesMaxAct`
  - `SID_MaxAct`
  - `ExtMoveAct`
  - `VlvPosMaxAct`
  - `VlvPosMinAct`
  - `CtrlDevMaxAct`

**Note**

Additional information about `Feature.Bit0` can be found in Set startup characteristics (Page 148)

### Status word assignment for the `Status1` parameter

For a description of the individual parameters, refer to the section I/Os of VlvMon (Page 582).

| Status bit | Parameter |
|---|---|
| 0 – 2 | `Not used` |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `OnAct.Value` |

| Status bit | Parameter |
|---|---|
| 6 – 10 | Not used |
| 11 | FlowAct.Value |
| 12 | FlowDriftAct.Value |
| 13 | RespUpAct.Value |
| 14 | RespDnAct.Value |
| 15 - 16 | Not used |
| 17 | T_MoveMaxAct.Value |
| 18 | T_IdleMaxAct.Value |
| 19 | DeadAct.Value (available in a later version) |
| 20 | StrokesMaxAct.Value |
| 21 | SID_MaxAct.Value |
| 22 | ExtMoveAct.Value |
| 23 | VlvPosMaxAct.Value |
| 24 | VlvPosMinAct.Value |
| 25 | CtrlDevMaxAct.Value |
| 26 | SuppPresAct.Value |
| 27 | Not used |
| 28 | Internal error |
| 29 – 31 | Not used |

### Status word assignment for `Status2` parameter

For a description of the individual parameters, refer to the section I/Os of VlvMon (Page 582).

| Status bit | Parameter |
|---|---|
| 0 | Not used |
| 1 | Feature.Bit10 |
| 2 | Feature.Bit11 |
| 3 | Feature.Bit12 |
| 4 - 31 | Not used |

## 4.3.2 VlvMon modes

### Operating modes of VlvMon

This block can be set to the following modes:

- On
- Out of service

**"On"**

General information about the "On" operating mode can be found in On (Page 77)

**"Out of service"**

General information about the "Out of service" operating mode can be found in Out of service (Page 77)

## 4.3.3 VlvMon operating principle

The VlvMon block is supplied with the current process values in the CFC via the inputs `MV` and `Rbk`. If the flow characteristic curve is to be used, the inputs `Flow`, `P_In` (pressure before the valve) and `P_Out` (pressure after the valve) are needed.

### Preprocessing

The process values are standardized to SI units via the *`Offs` and *`Fact` inputs. The normalized values are output again at the output. The status signals of the process values (quality code) are not evaluated but are instead output again at the output.

- `siFlow.Value = FlowOffs + Flowfact * Flow.Value`
  (normalize to $m^3/h$)
- `siP_In.Value = P_InOffs + P_InFact * P_In.Value`
  (normalize to bar)
- `siP_Out.Value = P_OutOffs + P_OutFact * P_Out.Value`
  (normalize to bar)

### Flow characteristic curve

Monitoring the flow characteristic curve is the heart of the VlvMon block. This is done by setting a set characteristic curve in the block for the flow depending on the valve position. This set characteristic curve can be entered manually, but it can also be taught automatically.

To diagnose caking or damage to the interior of the valve, this set characteristic curve is compared to the current operating point and an automatic acquired actual characteristic curve. Deviations from the set characteristic curve can be explained by a change to the valve cross-section.

This flow characteristic curve consists of 20 interpolation points. For the position feedback coming from the valve, the corresponding setpoint from the characteristic curve is calculated and compared to the actual flow.

#### (1) Normalization of the flow

To monitor the flow characteristic curve, the flow must be outside the influence of the surrounding pressure. Using the two pressure sensors upstream and downstream of the control valve, the flow is therefore standardized to a standard pressure difference. Otherwise, a change

in the pump power would result in a shift of the trend, for example. For the flow standardization $F_{Norm}$, the flow is divided by the square root of the pressure difference.

$$F_{Norm}(A) = \frac{F(A, \Delta p)}{\sqrt{\Delta p}} = \alpha A \sqrt{\frac{1}{\rho/2}}$$

| | |
|---|---|
| $F_{Norm}$ | *= normalized flow rate* |
| $F$ | *= flow rate* |
| $Q$ | *= valve cross-section* |
| $\Delta p$ | *= pressure drop* |
| $\rho$ | *= medium density* |
| $\alpha$ | *= volume flow coefficient* |

**(2) Operating point monitoring**

To diagnose damage to the interior of the valve, the set flow characteristic curve is compared to the current operating point (`siFlow.Value` and `Rbk.Value`) and an automatic acquired actual characteristic curve. If the deviation of the operating point from the set trend is within the defined tolerance (`FlowTol`), the operating point lights up green. Otherwise, excessive deviation is visualized by the pixel color switching to red.

Mechanical damage and caking inside the valve result in a change of the cross-section of the valve opening. The result is an altered flow with the same valve position. If the current operating point is below the characteristic curve, the current flow is too low for the valve position and, for example, caking or blockage is indicated. If, however, the operating point is above the characteristic curve, material is missing in the valve. This may be a sign of material loss due to cavitation, rust or a fracture in the valve plug. In addition to mechanical damage, flow monitoring can also be used for early detection of blockage and routing of solids.

Operating point monitoring thus detects a temporary deviation of the set characteristic curve and outputs both the absolute deviation (`DevFlow`) as well as the percentage deviation (`RelFlow`). When the maximum percentage deviation (`FlowTol`) is exceeded, the `FlowAct` output is activated.

**(3) Automatic characteristic curve detection**

Automatic characteristic curve detection has many advantages and opens up new possibilities for monitoring. It can be used to reduce the engineering workload required to commission the VlvMon block. Characteristic curve detection during ongoing operation also makes it easier to recognize trends.

The benefits it provides can be divided into two areas:

- Commissioning: Learning in the characteristic curve in good valve state and setting the automatically detected actual characteristic curve as the set characteristic curve.
- Monitoring during production: Acquiring the actual characteristic curve for drift detection (comparison of two characteristic curves).

The block monitors the various input variables. The new measured values are collected and form the basis of an actual characteristic curve which is as up-to-date as possible. The SteadyState block is used to detect steady measured values. (See SteadyState in the online help)

The automatically detected flow characteristic curve consists of 20 interpolation points (`ActFlow1...ActFlow20`, `ActVlvPos1...ActVlvPos20`). Each interpolation point is formed from the mean of up to 5 measured values. This mean value generation reduces the influence of individual outliers and measurement inaccuracies.

In addition, this mean value generation is weighted according to the up-to-dateness of the individual measured values. New measured values therefore have a stronger influence than old ones. Considering the up-to-dateness always ensures good drift detection. The up-to-dateness of the 20 interpolation points is displayed by the corresponding dot size in the faceplate. Larger dots here represent more up-to-date measured values. This allows better evaluation of the characteristic curve by the operator.

If a valve position has not yet been reached and there is thus no measured value for this interpolation point, it is calculated by linear interpolation from the surrounding interpolation points. Its up-to-dateness is thereby set to the minimum value by the system and the corresponding point is invisible in the faceplate.

If no interpolation points have been configured for the set characteristic curve, the current actual characteristic curve can be adopted as the set characteristic curve (`Adopt_ActFlow = 1`). Individual interpolation points of the actual characteristic curve can also be used to re-adjust the set characteristic curve. If the actual characteristic curve should be detected again, the current actual characteristic curve needs to be deleted (`RstActFlowOp = 1`).

**(4) Drift detection**

Automatic characteristic curve detection provides the basis for long-term drift detection. In contrast to the operating point monitoring, it also detects deviations relating to larger ranges of valve positions and not only to the current valve position.

Here, the set characteristic curve is compared to the automatically detected actual characteristic curve. When `Feature` Bit 13 "Drift calculation flow characteristic curve active" is set, the block calculates the absolute area between the two trends and returns the percentage deviation in relation to the set characteristic curve at the `RelFlowDrift` output. If the deviation between the two trends exceeds the configured tolerance `FlowDriftTol`, the `FlowDriftAct` output is activated.

## Reaction time characteristic curves

The reaction time corresponds to the time required for a valve to move to a desired valve position (except for a dead band). It depends on the step change of the setpoint. The greater the distance to be moved, the longer the reaction time. Slow valve motion results in a long reaction time. An increasing reaction time may therefore be a sign of increased friction in the valve's interior, for example. In addition, a leak in the compressed air supply results in slower valve movement. Damage to the valve drive can also be a cause for a change in valve reaction time.

Additionally, the reaction time depends on the valve's direction of movement. Different characteristic curves can result from positive steps compared to negative steps. This is explained by the fact that the valves are often operated by compressed air in one direction and by a spring

in the other direction. These two drive systems respond differently to errors and valve damage. This makes it necessary to include two different characteristic curves.

**Note**

Reaction times are determined only for setpoint changes, meaning that reaction time monitoring does not work for valve control by a higher-level controller (e.g. a flow controller) in automatic mode with constantly variable valve setpoints. In this case, a setpoint change must occasionally be specified in manual mode of the master controller in order to check the valve reaction time.

**(1) Operating point monitoring**

To diagnose damage of the valve drive or friction problems in the valve seal, the most recently detected positive and negative setpoint change as operating point (`T_RespUp` and `StepHiUp`; `T_RespDn` and `StepHiDn`) must be compared to the corresponding set reaction time characteristic curve. If this deviation from the set trend is within the defined tolerance, the operating point lights up green. Otherwise, excessive deviation is visualized by the pixel color switching to red.

Operating point monitoring detects a temporary deviation of the set characteristic curve and calculates both the absolute deviation (`DevRespUp`, `DevRespDn`) and the percentage deviation (`RelRespUp`, `RelRespDn`). When the maximum percentage deviation `RespTol` is exceeded, the `RespAct` output is activated.

**Note**

The calculated values for the relative and absolute deviation from the set characteristic curves are not shown in the faceplate.

**(2) Automatic characteristic curve detection**

Automatic characteristic curve detection is always active. The new measured values form the basis of an actual characteristic curve which is as up-to-date as possible. The automatically detected reaction time characteristic curves consist of 20 interpolation points:

- `ActStepHiUp1`...`ActStepHiUp20`, `ActRespUp1`...`ActRespUp20` for Open
- `ActStepHiDn1`...`ActStepHiDn20`, `ActRespDn1`...`ActRespDn20` for Close

Each interpolation point is formed from the mean of up to 5 measured values. This mean value generation reduces the influence of individual outliers and measurement inaccuracies.

If a setpoint step change has not yet been reached and there is therefore no measured value for this interpolation point, it is calculated by linear interpolation from the surrounding interpolation points.

If no interpolation points have been configured for the set characteristic curve, the current actual characteristic curve can be adopted as the set characteristic curve (`Adopt_ActResp = 1`). Individual interpolation points of the actual characteristic curve can also be used to re-adjust the

set characteristic curve. If the actual characteristic curve should be detected again, the current actual characteristic curve needs to be deleted (`RstActRespOp = 1`).

**Note**

The up-to-dateness of the measured interpolation points is not evaluated for the automatically detected reaction time characteristic curves.

**(3) Control deviation**

When the valve completes its motion after a setpoint change, its final position is compared to the set position. The difference represents the steady state control deviation. A steady state control deviation can have many different causes. Problems with the drive and the power supply are just as likely as temporary mechanical impairments.

If, after reaching the set position (PosReached = TRUE), the control deviation ER exceeds the configured tolerance `CtrlDevMax` for longer than the time specified by the `T_RespMax` parameter, the `CtrlDevMaxAct` output is enabled. Monitoring is reset as soon as a new setpoint change is detected.

## Monitoring functions

**(1) Maximum time in motion**

The block monitors the time in which the valve is in constant motion. For this, the maximum time for which the valve may move without a break is specified in [min] at the `T_MoveMax` parameter.

The current timer is sent to the `T_Move` output parameter. If the current timer exceeds the maximum value, the output parameter `T_MoveMaxAct` is activated.

The function can be deactivated with `T_MoveMax = 0`.

**(2) Maximum idle time**

The block monitors the time during which the valve remain still. For this, the maximum uninterrupted time for which the valve may remain still is specified in [min] at the `T_IdleMax` parameter.

The current timer is sent to the `T_Idle` output parameter. If the current timer exceeds the maximum value, the output parameter `T_IdleMaxAct` is activated.

The function can be deactivated with `T_IdleMax = 0`.

**(3) Valve strokes**

For the number of valve strokes, the distances traveled (%) with each valve movement (positive and negative) are added together. A stroke results from 100% valve movement. It is irrelevant whether the valve movement is performed only in the same range and in small steps. Movement from 0 to 50% and back is thus counted as one stroke, as well as 5 movements from 0 to 10% and back again.

If the number of strokes exceeds the maximum number set by the StrokesMax parameter, the output parameter StrokesMaxAct is activated.

The wear reserve is output in percent at the StrokesWearRes output parameter. The wear reserve is calculated as follows:

$$StrokesWearRes = \frac{StrokesMax - Strokes}{StrokesMax} \cdot 100$$

The function can be deactivated with StrokesMax = 0.

The input parameter RstStrokesOp = TRUE sets the Strokes = 0 parameter and the measurement starts again.

**(4) Direction change**

The current number of direction changes is stored in the SID parameter.

All changes of position feedback greater than the tolerance specified with the MoveTol parameter are tested for their effective direction and compared to the last known effective direction. If there is a difference, the direction change counter is increased by 1.

If the number of direction changes exceeds the maximum number set by the SID_Max parameter, the output parameter SID_MaxAct is activated.

The wear reserve is specified at the SID_WearRes output parameter. The wear reserve is calculated as follows:

$$SID_WearRes = \frac{SID_Max - SID}{SID_Max} \cdot 100$$

The function can be deactivated with SID_Max = 0.

SID = 0 is set with the input parameter RstSID_Op = TRUE.

**(5) External movement**

When the position setpoint MV no longer changes, the current feedback position is saved. If the position feedback Rbk of the valve then changes by more than the adjustable tolerance ExtMoveTol without another setpoint change (greater than MoveTol), the output parameter ExtMoveAct is activated.

The function can be deactivated with ExtMoveTol = 0.

**(6) High and low end stop**

The block monitors the valve for the high and low end stop. The position feedback Rbk is monitored here for the adjustable position limits VlvPosMin and VlvPosMax. If a limit is violated, the output parameter VlvPosMaxAct or VlvPosMinAct is activated.

The monitoring cannot be disabled.

The VlvPosHyst parameter can be used to specify the hysteresis for the limit violation.

The values for VlvPosMin and VlvPosMax must be specified during commissioning. For this, the respective end position is approached and the measured value of the position feedback is specified at the respective parameter. VlvPosMin and VlvPosMax then define the physical end positions at the time of commissioning.

**(7) Air supply pressure monitoring**

External air supply pressure monitoring can be interconnected at the SuppPres block input. An active signal (SuppPres = TRUE) is regarded as an error here. The signal in the block is only forwarded to the output parameter SuppPresAct. The signal is not processed further.

## Histograms

The histograms show the frequency distributions of the various valve positions or setpoint step changes over the entire travel range (0 to 100%) of the valve position as a bar graph.

The RstHisOp reset input is used to reset the counters and the total counters T_HisPos, CntHisStepUp and CntHisStepDn to zero and restart them.

**(1) Histogram for valve position**

This histogram shows the frequency of the various valve positions over the full range of the valve movement.

The valve position range 0...100% is divided into 10 equal sub-ranges. Each of these sub-ranges is assigned a counter for the duration of stay:

- HisPos1 corresponds to >0-10% valve position
- HisPos2 corresponds to 10-20% valve position
- HisPos3 corresponds to 20-30% valve position
- HisPos4 corresponds to 30-40% valve position
- HisPos5 corresponds to 40-50% valve position
- HisPos6 corresponds to 50-60% valve position
- HisPos7 corresponds to 60-70% valve position
- HisPos8 corresponds to 70-80% valve position
- HisPos9 corresponds to 80-90% valve position
- HisPos10 corresponds to 90-100% valve position

There is also a counter to indicate the "closed" status.

- HisPos0 corresponds to the "closed" state (FbkCloseOut.Value = TRUE)

The T_HisPos counter is incremented with the sampling time and specifies the monitoring duration in [h]. The individual counters are incremented on the basis of the position feedback. They indicate the operating time of the component in [%] of the recording period.

**Evaluation of the histogram:**

Valves should be operated predominantly in the middle section (10-90%) of their actuator travel, not at the extremes.

- If you notice strong clusters in a range of very small openings, the dimensions of the valve are too large. In a range of very small openings, the valve characteristic curve is not reproducible, making control difficult, and there is a danger of cavitation. In addition, an almost completely closed valve is associated with greater pressure loss due to the deflection and the flow through the narrow openings.
- If you notice strong clusters in a range of very large openings, the dimensions of the valve are too small. This means that a flow controller acting upon this with its manipulated variable has little room to affect positive setpoint changes. If the valve in its fully open state actually has a smaller cross-section than that of the pipe, eddying and pressure losses occur at the joint.

**(2) Histogram for positive and negative step changes**

This histogram shows the frequency of the various setpoint step changes over the complete range of possible positive and negative steps.

**Positive valve steps:**

The valve position range `0...100%` is divided into 10 equal sub-ranges. Each of these sub-ranges is assigned a counter for the corresponding step change:

- `HisStepHiUp1` corresponds to step change 0-10%
- `HisStepHiUp2` corresponds to step change 10-20%
- `HisStepHiUp3` corresponds to step change 20-30%
- `HisStepHiUp4` corresponds to step change 30-40%
- `HisStepHiUp5` corresponds to step change 40-50%
- `HisStepHiUp6` corresponds to step change 50-60%
- `HisStepHiUp7` corresponds to step change 60-70%
- `HisStepHiUp8` corresponds to step change 70-80%
- `HisStepHiUp9` corresponds to step change 80-90%
- `HisStepHiUp10` corresponds to step change 90-100%

The `CntHisStepUp` counter is incremented with each positive step detected and indicates the total number of positive steps. The individual counters are incremented on the basis of the current step change. They provide an overview of the distribution of positive steps in relation to the total number of steps.

**Negative valve steps:**

The valve position range `100...0%` is divided into 10 equal sub-ranges. Each of these sub-ranges is assigned a counter for the corresponding step change:

- `HisStepHiDn1` corresponds to step change 0 to -10%
- `HisStepHiDn2` corresponds to step change -10 to -20%
- `HisStepHiDn3` corresponds to step change -20 to -30%
- `HisStepHiDn4` corresponds to step change -30 to -40%

- `HisStepHiDn5` corresponds to step change -40 to -50%
- `HisStepHiDn6` corresponds to step change -50 to -60%
- `HisStepHiDn7` corresponds to step change -60 to -70%
- `HisStepHiDn8` corresponds to step change -70 to -80%
- `HisStepHiDn9` corresponds to step change -80 to -90%
- `HisStepHiDn10` corresponds to step change -90 to -100%

The `CntHisStepDn` counter is incremented with each negative step detected and indicates the total number of negative steps. The individual counters are incremented on the basis of the current step change. They provide an overview of the distribution of negative steps in relation to the total number of steps.

### 4.3.4 VlvMon functions

The functions for this block are listed below.

**Analyses and characteristic curve displays of the valve data and operating states**

- Set flow characteristic curve: generated by applying the actual characteristic measured automatically after commissioning. Display of the set flow depending on the valve position with standardized flow (i.e. flow converted to standard differential pressure)
- Display of the current operating point, the absolute and percentage deviation of the operating point from the set characteristic curve.
- Actual flow characteristic curve: Determined automatically during ongoing operation. Display of the actual flow depending on the valve position with standardized flow, enables the detection of characteristic curve shifts, calculation of the percentage deviation of the actual characteristic curve compared to the set characteristic curve.
- Set reaction time characteristic curves: generated by applying the actual characteristic measured automatically after commissioning. Display of the set valve reaction time depending on the step change for positive and negative setpoint changes
- Display of the last detected positive or negative setpoint change and calculation of the absolute and percentage deviation of the step changes of the set characteristic curve.
- Actual reaction time characteristic curves: Display of the actual valve reaction time depending on the step change for positive and negative setpoint changes; enables the detection of characteristic curve shifts.
- Histogram (frequency distribution of the valve settings): Statistical evaluation of valve positions.

### Diagnostic functions for each control valve with feedback of the valve position

The block offers the following basic diagnostic functions to warn operating personnel of adverse operating conditions or of wear limits being reached. These diagnostic functions do not detect valve errors in the strict sense. However, they do help prevent valve failure. They allow for better maintenance planning and detect when a valve has exceeded its expected service life:

- Monitoring the max. allowable operating hours in continuous standstill without valve movement → caking, encrustation.
- Monitoring the max. allowable operating hours in continuous operation without valve standstill → overload.
- Monitoring the max. allowable number of valve strokes → maintenance.
- Monitoring the max. allowable number of direction changes → maintenance.

The block also recognizes a faulty valve movement using the following diagnostic procedures. Here, the setpoint given by the positioner is compared to the actual feedback of the valve. This can be used to detect wear, caking, sluggishness, increased friction and other valve movement errors:

- Detection of valve movements without position commands → air pressure leakage or external mechanical influences.
- Shifting of the high and low end position → damage to the valve plug, point of origin shifting.
- Determination of a permanent control deviation after the valve movement → indication of problems in the drive, supply voltage or the air supply.
- Monitoring of the valve response time with reference to a set response time characteristic curve for each valve movement → sluggishness, damage to the valve drive.

### Advanced diagnostic functions with additional sensors

With additional sensors that may already be installed in the valve environment, the block can perform the following advanced diagnostic functions:

*- With additional pressure transducer on inlet and outlet side as well as flow sensor:*

- Operating point monitoring of the flow characteristic curve based on a set flow characteristic curve depending on the valve position → information on changing the valve cross-section due to caking or abrasion.
- Comparison of the automatically detected actual flow characteristic curve with the set flow characteristic curve for long-term drift detection → mechanical damage to the valve plug, caking, abrasion.

*- With additional supply compressed air sensor:*

- Monitoring the air supply pressure → leak in the compressed air supply.

### Teach functions

- Step-by-step learning of the set flow characteristic curve
- Step-by-step learning of the set reaction time characteristic curves

### Application cases / monitoring candidates

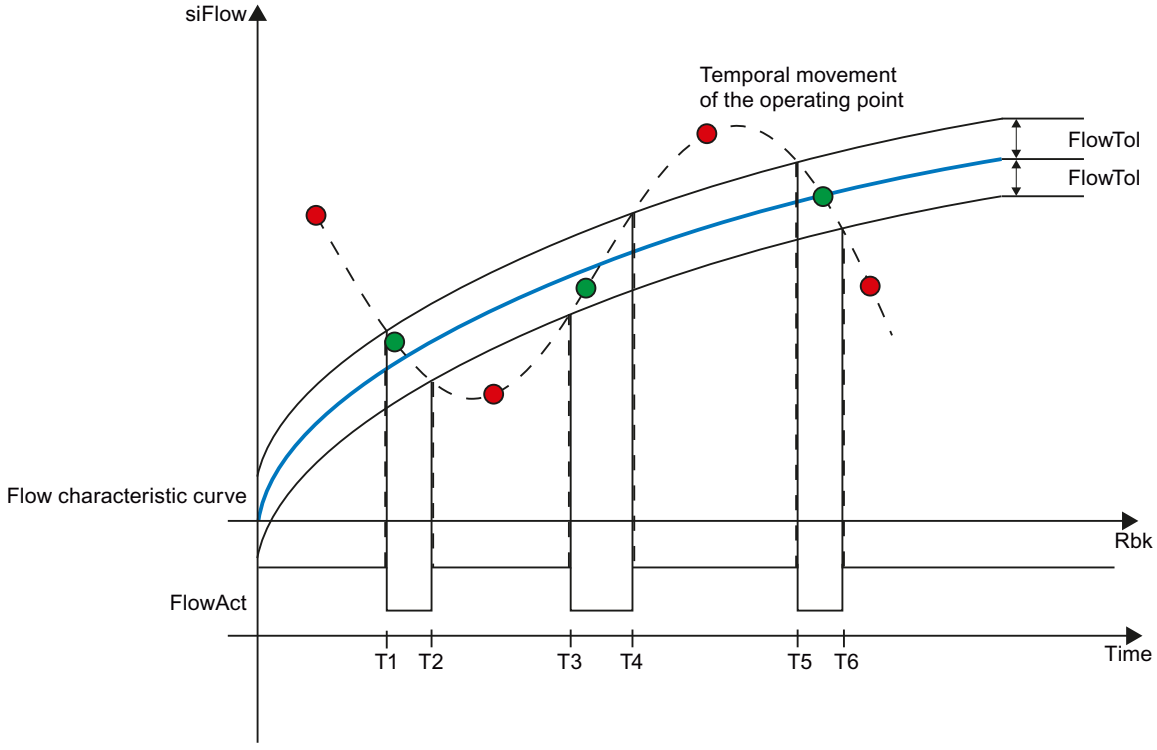
- Control valves that demonstrate above-average occurrence of malfunctions and wear.
- Control valves in which chemical deposits form.
- Control valves that have an inexplicable deviation with respect to its original flow and reaction time characteristic curves.

### Tolerance monitoring of the characteristic curves

The block monitors the deviation of the valve operating point from the flow characteristic curve and the reaction time characteristic curves for adjustable tolerances.

The functionality of the monitoring is based on the standard Limit monitoring (Page 93).

Monitoring the flow characteristic curve involves checking whether the operating point is within the tolerance range (see figure). If it is outside the tolerance range, the binary output variable FlowAct (i.e. "Flow Deviation Active") is set.




The characteristic curve monitoring for

- "Open" response time
- "Close" response time

checks whether the operating point, which is formed the most recently detected step, is within the tolerance range (see figure).

If the operating point leaves the tolerance range, the operating point in the faceplate changes its color from green to red and the corresponding output parameter is activated (`FlowAct`, `RespUpAct`, `RespDnActAct`).

## Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The signal status of the FbkCloseOut and FbkOpenOut outputs is formed from the worst signal status of the Rbk output and the corresponding feedback signal input FbkClose or FbkOpen and output.

The signal status of the valve position control deviation ER always corresponds to the signal status of the Rbk input.

The worst signal status ST_Worst for the block is formed from the following parameters:

- Rbk.ST
- Flow.ST
- P_In.ST
- P_Out.ST
- MV.ST
- FbkOpenOut.ST
- FbkCloseOut.ST

The signal states of the corresponding input measured values are used to form the signal status at the output end:

| Signal status | Derived from the signal status of the input variable |
|---|---|
| siFlow.ST | Flow.ST |
| siP_In.ST | P_In.ST |
| siP_Out.ST | P_Out.ST |

## Monitoring of feedback

This block provides the standard function Monitoring the feedbacks (Page 104).

Digital feedback signals for the "Open" and "Closed" positions are formed from the position feedback:

- Feedback for "Open" position: Rbk ≥ PosDiOpen
- Feedback for "Closed" position: Rbk ≤ PosDiClose

**Specifying control valve positions:**

There are the following positions or travel states:

- Valve closed ("Closed")
- Valve closes
- Valve open ("Open")
- Valve opens
- Valve has reached defined position

**Valve closed ("Closed")**

When the control valve reaches the "Close" position, `FbkCloseOut.Value = 1`:

- With a binary limit switch for the "Closed" position (`NoFbkClose = 0`):
  The valve is considered closed when `FbkCloseOut` is set. `FbkCloseOut` is set when `Rbk` ≤ `PosDiClose` and `FbkClose = 1`.
- Without a binary limit switch for the "Closed" position (`NoFbkClose = 1`):
  The valve is considered closed when FbkCloseOut is set. FbkCloseOut is set when Rbk ≤ PosDiClose.

**Valve closes**

If the control valve travels in the direction of the "Close" position, the `FbkClsgOut = 1` output is:

- `FbkClsgOut` is set when `MV.Value` < `RbkOut.Value` and not `PosReached.Value = 1`.

**Valve open ("Open")**

When the control valve reaches the "Open" position, the `FbkOpenOut.Value = 1` output is:

- With a binary limit switch for the "Open" position (`NoFbkOpen= 0`):
  The valve is considered open when `FbkOpenOut` is set. `FbkOpenOut` is set when `Rbk` ≥ `PosDiOpen` and `FbkOpen = 1`.
- Without a digital limit switch for the "Open" position (`NoFbkOpen = 1`):
  The valve is considered open when `FbkOpenOut` is set. `FbkOpenOut` is set when `Rbk` ≥ `PosDiOpen`.

**Valve opens**

If the control valve travels in the direction of the "Open" position, the output `FbkOpngOut = 1`:

- `FbkOpngOut` is set when `MV.Value` > `RbkOut.Value` and not `PosReached.Value = 1`.

**Valve has reached defined position**

If the control valve travels to a defined intermediate position (`MV >PosDiClose` and `MV<PosDiOpen`), the targeted position is reached when the difference `MV.Value`– `RbkOut.Value` is within the configured tolerance range ± `PosDeadBand` and therefore `ER.Value=0.0`.

If the control valve is set to the end position "Open" (`MV.Value >= PosDiOpen`), this position is reached when `ER.Value = 0.0` and also `FbkOpenOut.Value = 1`.

If the control valve is set to the end position "Open" (`MV.Value<= PosDiClose`), this position is reached when `ER.Value = 0.0` and also `FbkCloseOut.Value = 1`.

When the control valve reaches the specified position, the output `PosReached = 1` is set.

## Deactivate feedback

This block provides the standard function Disabling feedback for valves (Page 106). Feedback monitoring can be disabled separately for each feedback signal with `NoFbkOpen` or `NoFbkClose`.

### Selecting a unit of measurement

This block provides the standard function Selecting a unit of measure (Page 240).

### Configurable reactions using the "Feature" parameter

An overview of all the reactions that are provided by the `Feature` parameter is available in section Configurable response using the Feature I/O (Page 139). The following modes of behavior are available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Setting the startup characteristics |
| 1 | Characteristics for the out of service mode |
| 10 | Flow characteristic curve active |
| 11 | "Open" response time characteristic curve active |
| 12 | "Close" response time characteristic curve active |
| 13 | Drift calculation flow characteristic curve active |
| 26 | Behavior of the switching points in "Out of service" mode |
| 28 | Disabling of switching points |
| 29 | Signaling limit violation |

### Operator control permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 – 5 | Not used |
| 6 | 1 = Operator can reset the histogram |
| 7 – 9 | Not used |
| 10 | 1 = Operator can adopt the current flow actual characteristic curve |
| 11 | 1 = Operator can reset the current flow actual characteristic curve |
| 12 | Not used |
| 13 | Not used |
| 14 | Not used |
| 15 | 1 = Operator can adopt the current reaction time actual characteristic curve |
| 16 | 1 = Operator can reset the current reaction time actual characteristic curve |
| 17 | Not used |
| 18 | Not used |
| 19 | 1 = Operator can reset the change of direction counter |

| Bit | Function |
|---|---|
| 20 | 1 = Operator can reset the stroke counter |
| 21 | Not used |
| 22 | Not used |
| 23 | 1 = Operator can change the characteristic curve values for the valve position |
| 24 | 1 = Operator can change the characteristic curve values for the flow |
| 25 | 1 = Operator can change the characteristic curve values for positive steps |
| 26 | 1 = Operator can change the characteristic curve values for negative steps |
| 27 | 1 = Operator can change the characteristic curve values for the reaction time for opening |
| 28 | 1 = Operator can change the characteristic curve value for the reaction time for closing |
| 29 | 1 = Operator can change the maximum reaction time |
| 30 – 31 | Not used |

The block has the following permissions for the `OS1Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can change the limit for detection of a movement |
| 1 | 1 = Operator can change the limit for detection of the end position |
| 2 | 1 = Operator can change the limit for detection of an external movement |
| 3 | Not used |
| 4 | 1 = Operator can change the limit for the maximum time in motion |
| 5 | 1 = Operator can change the limit for the maximum idle time |
| 6 | 1 = Operator can change the limit for the maximum number of strokes |
| 7 | 1 = Operator can change the limit for the maximum number of changes of direction |
| 8 | Not used |
| 9 | Not used |
| 10 | Not used |
| 11 | Not used |
| 12 | Not used |
| 13 | Not used |
| 14 | Not used |
| 15 | 1 = Operator can change the limit for the deviation of the flow characteristic curve |
| 16 | 1 = Operator can change the limit for the deviation of the reaction time characteristic curve |
| 17 | Not used |
| 18 | Not used |
| 19 | Not used |
| 20 | Not used |
| 21 – 31 | Not used |

### Opening additional faceplates

You can open the standard view of the technological block (e.g. VlvAnL) interconnected to the input parameter MV using a button in the standard view of the faceplate. The interconnection makes the button in the standard view visible.

### SIMATIC BATCH functionality

This block does not support "SIMATIC BATCH functionality".

## 4.3.5 VlvMon error handling

The block overwrites the following parameters with 0 if they were configured with a negative value:

- FlowTol
- RespTol
- FlowCurveTol
- MoveTol
- ExtMoveTol
- DeadTol
- FinalTol
- StrokesMax
- SID_Max
- StepHiMax
- T_RespMax
- T_IdleMax
- T_MoveMax
- CtrlDevMax
- VlvPosHyst

### Overview of error numbers

The ErrorNum I/O can be used to output various error numbers:

| Error ID | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |

## 4.3.6 VlvMon messaging

### Message behavior

The VlvMon block does not generate messages or maintenance alarms. It simply provides several binary message outputs that are set when the permissible deviation is exceeded.

Messages may be generated for specific applications by a separate event block, and maintenance alarms by a downstream AssetMon block (included in the standard package of PCS 7). Events that are relevant for the plant operator can also be reported via interconnection of VlvMon message outputs to free message inputs of the associated VlvAnL.

## 4.3.7 I/Os of VlvMon

### I/Os of VlvMon

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Adopt_ActFlow | 1 = Adopt the current actual flow rate characteristic curve | BOOL | FALSE |
| Adopt_ActResp | 1 = Adopt the current actual response time characteristic curves | BOOL | FALSE |
| CtrlDevMax | Maximum control deviation | REAL | 0.0 |
| CycleTime | Time period in which a new measured value is used for calculation per interpolation point | TIME | T#5m |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| DeadActIn | 1 = Dead time monitoring active (available in a later version) | BOOL | FALSE |
| DeadTol | Tolerance for dead time monitoring (available in a later version) | REAL | 0.0 |
| ExtMoveTol | Tolerance for detection of an external movement | REAL | 0.0 |
| FbkClose | "1 = Valve closed feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| FbkOpen | signal" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Feature | 1 = Valve open feedback signal | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| FinalTol | Tolerance for detecting the end position after a movement | REAL | 1.0 |
| Flow | Interconnectable input for the flow | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Flow1 | y value 1, flow [m³/h] | REAL | 1.0 |
| Flow10 | y value 10, flow [m³/h] | REAL | 100.0 |
| Flow11 | y value 11, flow [m³/h] | REAL | 121.0 |
| Flow12 | y value 12, flow [m³/h] | REAL | 144.0 |
| Flow13 | y value 13, flow [m³/h] | REAL | 169.0 |
| Flow14 | y value 14, flow [m³/h] | REAL | 196.0 |
| Flow15 | y value 15, flow [m³/h] | REAL | 225.0 |
| Flow16 | y value 16, flow [m³/h] | REAL | 256.0 |
| Flow17 | y value 17, flow [m³/h] | REAL | 289.0 |
| Flow18 | y value 18, flow [m³/h] | REAL | 324.0 |
| Flow19 | y value 19, flow [m³/h] | REAL | 361.0 |
| Flow2 | y value 2, flow [m³/h] | REAL | 4.0 |
| Flow20 | y value 20, flow [m³/h] | REAL | 400.0 |
| Flow3 | y value 3, flow [m³/h] | REAL | 9.0 |
| Flow4 | y value 4, flow [m³/h] | REAL | 16.0 |
| Flow5 | y value 5, flow [m³/h] | REAL | 25.0 |
| Flow6 | y value 6, flow [m³/h] | REAL | 36.0 |
| Flow7 | y value 7, flow [m³/h] | REAL | 49.0 |
| Flow8 | y value 8, flow [m³/h] | REAL | 64.0 |
| Flow9 | y value 9, flow [m³/h] | REAL | 81.0 |
| FlowDriftTol | Maximum deviation between the flow rate set and actual characteristic | REAL | 5.0 |
| FlowFact | Flow factor, normalization to m³/h | REAL | 1.0 |
| FlowMax | Maximum flow (indicates the maximum value for the y-axis) [m³/h] | REAL | 500.0 |
| FlowOffs | Flow offset, normalization to m³/h | REAL | 0.0 |
| FlowStat | Interconnectable input for the stationary state monitoring of the flow (SteadyState) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FlowTol | Maximum deviation from the flow characteristic curve | REAL | 5.0 |
| FlowUnit | Unit for the flow | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MoveTol | Tolerance for detecting a movement | REAL | 1.0 |
| MV | Manipulated variable (via interconnection) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_Unit | Unit of measurement for manipulated variable | INT | 1342 |
| NoFbkClose | 1 = No feedback present for "control valve closed" | BOOL | FALSE |
| NoFbkOpen | 1 = No feedback present for "control valve open" | BOOL | FALSE |
| OnOp | 1 = "On", via OS operator input | BOOL | FALSE |
| OosLi | 1 = "Out of service", via interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator input | BOOL | FALSE |
| OS_Perm | I/O for operator control permissions (see VlvMon functions (Page 574)) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| OS1Perm | I/O for operator control permissions (see VlvMon functions (Page 574)) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| P_In | Interconnectable input for the input pressure | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| P_InFact | Input pressure factor, normalization to bar | REAL | 1.0 |
| P_InOffs | Input pressure offset, normalization to bar | REAL | 0.0 |
| P_InUnit | Unit of measure for the input pressure | INT | 0 |
| P_Out | Interconnectable input for the output pressure | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| P_OutFact | Output pressure factor, normalization to bar | REAL | 1.0 |
| P_OutOffs | Output pressure offset, normalization to bar | REAL | 0.0 |
| P_OutUnit | Unit of measure for output pressure | INT | 0 |
| PosDeadBand | Dead band for forming manipulated variable difference | REAL | 0.1 |
| PosDiClose | Limit for control valve position "Closed" | REAL | 2.0 |
| PosDiOpen | Limit for control valve position "Open" | REAL | 98.0 |
| Rbk | Interconnectable input for the position feedback | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RbkUnit | Unit of measurement for the position feedback | INT | 1342 |
| RespDn1 | y value 1, close response time [s] | REAL | 0.25 |
| RespDn10 | y value 10, close response time [s] | REAL | 2.5 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RespDn11 | y value 11, close response time [s] | REAL | 2.75 |
| RespDn12 | y value 12, close response time [s] | REAL | 3.0 |
| RespDn13 | y value 13, close response time [s] | REAL | 3.25 |
| RespDn14 | y value 14, close response time [s] | REAL | 3.5 |
| RespDn15 | y value 15, close response time [s] | REAL | 3.75 |
| RespDn16 | y value 16, close response time [s] | REAL | 4.0 |
| RespDn17 | y value 17, close response time [s] | REAL | 4.25 |
| RespDn18 | y value 18, close response time [s] | REAL | 4.5 |
| RespDn19 | y value 19, close response time [s] | REAL | 4.75 |
| RespDn2 | y value 2, close response time [s] | REAL | 0.5 |
| RespDn20 | y value 20, close response time [s] | REAL | 5.0 |
| RespDn3 | y value 3, close response time [s] | REAL | 0.75 |
| RespDn4 | y value 4, close response time [s] | REAL | 1.0 |
| RespDn5 | y value 5, close response time [s] | REAL | 1.25 |
| RespDn6 | y value 6, close response time [s] | REAL | 1.5 |
| RespDn7 | y value 7, close response time [s] | REAL | 1.75 |
| RespDn8 | y value 8, close response time [s] | REAL | 2.0 |
| RespDn9 | y value 9, close response time [s] | REAL | 2.25 |
| RespTol | Maximum deviation from the characteristic curves | REAL | 5.0 |
| RespUp1 | y value 1, open response time [s] | REAL | 0.5 |
| RespUp10 | y value 10, open response time [s] | REAL | 5.0 |
| RespUp11 | y value 11, open response time [s] | REAL | 5.5 |
| RespUp12 | y value 12, open response time [s] | REAL | 6.0 |
| RespUp13 | y value 13, open response time [s] | REAL | 6.5 |
| RespUp14 | y value 14, open response time [s] | REAL | 7.0 |
| RespUp15 | y value 15, open response time [s] | REAL | 7.5 |
| RespUp16 | y value 16, open response time [s] | REAL | 8.0 |
| RespUp17 | y value 17, open response time [s] | REAL | 8.5 |
| RespUp18 | y value 18, open response time [s] | REAL | 9.0 |
| RespUp19 | y value 19, open response time [s] | REAL | 9.5 |
| RespUp2 | y value 2, open response time [s] | REAL | 1.0 |
| RespUp20 | y value 20, open response time [s] | REAL | 10.0 |
| RespUp3 | y value 3, open response time [s] | REAL | 1.5 |
| RespUp4 | y value 4, open response time [s] | REAL | 2.0 |
| RespUp5 | y value 5, open response time [s] | REAL | 2.5 |
| RespUp6 | y value 6, open response time [s] | REAL | 3.0 |
| RespUp7 | y value 7, open response time [s] | REAL | 3.5 |
| RespUp8 | y value 8, open response time [s] | REAL | 4.0 |
| RespUp9 | y value 9, open response time [s] | REAL | 4.5 |
| RstActFlowOp* | 1 = Reset actual flow characteristic curve | BOOL | FALSE |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RstActRespOp* | 1 = Reset actual response time characteristic curves | BOOL | FALSE |
| RstHisOp* | 1 = Reset histograms | BOOL | FALSE |
| RstSID_Op* | 1 = Reset direction change counter | BOOL | FALSE |
| RstStrokesOp* | 1 = Reset stroke counter | BOOL | FALSE |
| RunUpCyc | Number of cycles during startup (not currently evaluated) | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SID_Max | Indicates the maximum number of direction changes | DINT | 0 |
| siFlowUnit | Unit of measure for the normalized flow | INT | 1349 |
| siP_Unit | Unit of measure for the normalized pressure | INT | 1137 |
| StepHiDn1 | x value 1, negative step change [%] | REAL | 5.0 |
| StepHiDn10 | x value 10, negative step change [%] | REAL | 50.0 |
| StepHiDn11 | x value 11, negative step change [%] | REAL | 55.0 |
| StepHiDn12 | x value 12, negative step change [%] | REAL | 60.0 |
| StepHiDn13 | x value 13, negative step change [%] | REAL | 65.0 |
| StepHiDn14 | x value 14, negative step change [%] | REAL | 70.0 |
| StepHiDn15 | x value 15, negative step change [%] | REAL | 75.0 |
| StepHiDn16 | x value 16, negative step change [%] | REAL | 80.0 |
| StepHiDn17 | x value 17, negative step change [%] | REAL | 85.0 |
| StepHiDn18 | x value 18, negative step change [%] | REAL | 90.0 |
| StepHiDn19 | x value 19, negative step change [%] | REAL | 95.0 |
| StepHiDn2 | x value 2, negative step change [%] | REAL | 10.0 |
| StepHiDn20 | x value 20, negative step change [%] | REAL | 100.0 |
| StepHiDn3 | x value 3, negative step change [%] | REAL | 15.0 |
| StepHiDn4 | x value 4, negative step change [%] | REAL | 20.0 |
| StepHiDn5 | x value 5, negative step change [%] | REAL | 25.0 |
| StepHiDn6 | x value 6, negative step change [%] | REAL | 30.0 |
| StepHiDn7 | x value 7, negative step change [%] | REAL | 35.0 |
| StepHiDn8 | x value 8, negative step change [%] | REAL | 40.0 |
| StepHiDn9 | x value 9, negative step change [%] | REAL | 45.0 |
| StepHiMax | Maximum step change (indicates the maximum value for the x-axes) | REAL | 100.0 |
| StepHiUp1 | x value 1, positive step change [%] | REAL | 5.0 |
| StepHiUp10 | x value 10, positive step change [%] | REAL | 50.0 |
| StepHiUp11 | x value 11, positive step change [%] | REAL | 55.0 |
| StepHiUp12 | x value 12, positive step change [%] | REAL | 60.0 |
| StepHiUp13 | x value 13, positive step change [%] | REAL | 65.0 |
| StepHiUp14 | x value 14, positive step change [%] | REAL | 70.0 |
| StepHiUp15 | x value 15, positive step change [%] | REAL | 75.0 |
| StepHiUp16 | x value 16, positive step change [%] | REAL | 80.0 |
| StepHiUp17 | x value 17, positive step change [%] | REAL | 85.0 |
| StepHiUp18 | x value 18, positive step change [%] | REAL | 90.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| StepHiUp19 | x value 19, positive step change [%] | REAL | 95.0 |
| StepHiUp2 | x value 2, positive step change [%] | REAL | 10.0 |
| StepHiUp20 | x value 20, positive step change [%] | REAL | 100.0 |
| StepHiUp3 | x value 3, positive step change [%] | REAL | 15.0 |
| StepHiUp4 | x value 4, positive step change [%] | REAL | 20.0 |
| StepHiUp5 | x value 5, positive step change [%] | REAL | 25.0 |
| StepHiUp6 | x value 6, positive step change [%] | REAL | 30.0 |
| StepHiUp7 | x value 7, positive step change [%] | REAL | 35.0 |
| StepHiUp8 | x value 8, positive step change [%] | REAL | 40.0 |
| StepHiUp9 | x value 9, positive step change [%] | REAL | 45.0 |
| StrokesMax | Indicates the maximum number of complete valve strokes | DINT | 0 |
| SuppPres | Connected to air supply pressure monitoring; 1 = Error | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| T_IdleMax | Indicates the maximum time for continuous standstill | DINT | 0 |
| T_MoveMax | Indicates the maximum time for continuous motion | DINT | 0 |
| T_RespMax | Maximum response time (indicates the maximum value for the y-axis) | REAL | 20.0 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |
| VlvPos1 | x value 1, valve position | REAL | 5.0 |
| VlvPos10 | x value 10, valve position | REAL | 50.0 |
| VlvPos11 | x value 11, valve position | REAL | 55.0 |
| VlvPos12 | x value 12, valve position | REAL | 60.0 |
| VlvPos13 | x value 13, valve position | REAL | 65.0 |
| VlvPos14 | x value 14, valve position | REAL | 70.0 |
| VlvPos15 | x value 15, valve position | REAL | 75.0 |
| VlvPos16 | x value 16, valve position | REAL | 80.0 |
| VlvPos17 | x value 17, valve position | REAL | 85.0 |
| VlvPos18 | x value 18, valve position | REAL | 90.0 |
| VlvPos19 | x value 19, valve position | REAL | 95.0 |
| VlvPos2 | x value 2, valve position | REAL | 10.0 |
| VlvPos20 | x value 20, valve position | REAL | 100.0 |
| VlvPos3 | x value 3, valve position | REAL | 15.0 |
| VlvPos4 | x value 4, valve position | REAL | 20.0 |
| VlvPos5 | x value 5, valve position | REAL | 25.0 |
| VlvPos6 | x value 6, valve position | REAL | 30.0 |
| VlvPos7 | x value 7, valve position | REAL | 35.0 |
| VlvPos8 | x value 8, valve position | REAL | 40.0 |
| VlvPos9 | x value 9, valve position | REAL | 45.0 |
| VlvPosHyst | Indicates the hysteresis of the valve position [%] | REAL | 0.1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| VlvPosMax | Indicates the feedback value at the upper stop of the valve at the time of commissioning | REAL | 100.0 |
| VlvPosMin | Indicates the feedback value at the bottom stop of the valve at the time of commissioning | REAL | 0.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AccFlowPoint 1 | y value 1, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint 2 | y value 2, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint 3 | y value 3, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint 4 | y value 4, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint 5 | y value 5, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint 6 | y value 6, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint 7 | y value 7, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint 8 | y value 8, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint 9 | y value 9, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint 10 | y value 10, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint 11 | y value 11, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint 12 | y value 12, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint 13 | y value 13, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint 14 | y value 14, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint 15 | y value 15, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint 16 | y value 16, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint 17 | y value 17, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint 18 | y value 18, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| AccFlowPoint19 | y value 19, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| AccFlowPoint20 | y value 20, up-to-dateness of the actual flow characteristic curve | REAL | 1.0 |
| ActFlow1 | y value 1, actual flow characteristic curve | REAL | 1.0 |
| ActFlow2 | y value 2, actual flow characteristic curve | REAL | 2.0 |
| ActFlow3 | y value 3, actual flow characteristic curve | REAL | 3.0 |
| ActFlow4 | y value 4, actual flow characteristic curve | REAL | 4.0 |
| ActFlow5 | y value 5, actual flow characteristic curve | REAL | 5.0 |
| ActFlow6 | y value 6, actual flow characteristic curve | REAL | 6.0 |
| ActFlow7 | y value 7, actual flow characteristic curve | REAL | 7.0 |
| ActFlow8 | y value 8, actual flow characteristic curve | REAL | 8.0 |
| ActFlow9 | y value 9, actual flow characteristic curve | REAL | 9.0 |
| ActFlow10 | y value 10, actual flow characteristic curve | REAL | 10.0 |
| ActFlow11 | y value 11, actual flow characteristic curve | REAL | 11.0 |
| ActFlow12 | y value 12, actual flow characteristic curve | REAL | 12.0 |
| ActFlow13 | y value 13, actual flow characteristic curve | REAL | 13.0 |
| ActFlow14 | y value 14, actual flow characteristic curve | REAL | 14.0 |
| ActFlow15 | y value 15, actual flow characteristic curve | REAL | 15.0 |
| ActFlow16 | y value 16, actual flow characteristic curve | REAL | 16.0 |
| ActFlow17 | y value 17, actual flow characteristic curve | REAL | 17.0 |
| ActFlow18 | y value 18, actual flow characteristic curve | REAL | 18.0 |
| ActFlow19 | y value 19, actual flow characteristic curve | REAL | 19.0 |
| ActFlow20 | y value 20, actual flow characteristic curve | REAL | 20.0 |
| ActRespDn1 | y value 1, actual close response time characteristic curve [s] | REAL | 0.0 |
| ActRespDn2 | y value 2, actual close response time characteristic curve [s] | REAL | 1.0 |
| ActRespDn3 | y value 3, actual close response time characteristic curve [s] | REAL | 2.0 |
| ActRespDn4 | y value 4, actual close response time characteristic curve [s] | REAL | 3.0 |
| ActRespDn5 | y value 5, actual close response time characteristic curve [s] | REAL | 4.0 |
| ActRespDn6 | y value 6, actual close response time characteristic curve [s] | REAL | 5.0 |
| ActRespDn7 | y value 7, actual close response time characteristic curve [s] | REAL | 6.0 |
| ActRespDn8 | y value 8, actual close response time characteristic curve [s] | REAL | 7.0 |
| ActRespDn9 | y value 9, actual close response time characteristic curve [s] | REAL | 8.0 |
| ActRespDn10 | y value 10, actual close response time characteristic curve [s] | REAL | 9.0 |
| ActRespDn11 | y value 11, actual close response time characteristic curve [s] | REAL | 10.0 |
| ActRespDn12 | y value 12, actual close response time characteristic curve [s] | REAL | 11.0 |
| ActRespDn13 | y value 13, actual close response time characteristic curve [s] | REAL | 12.0 |
| ActRespDn14 | y value 14, actual close response time characteristic curve [s] | REAL | 13.0 |
| ActRespDn15 | y value 15, actual close response time characteristic curve [s] | REAL | 14.0 |
| ActRespDn16 | y value 16, actual close response time characteristic curve [s] | REAL | 15.0 |
| ActRespDn17 | y value 17, actual close response time characteristic curve [s] | REAL | 16.0 |
| ActRespDn18 | y value 18, actual close response time characteristic curve [s] | REAL | 17.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ActRespDn19 | y value 19, actual close response time characteristic curve [s] | REAL | 18.0 |
| ActRespDn20 | y value 20, actual close response time characteristic curve [s] | REAL | 19.0 |
| ActRespUp1 | y value 1, actual open response time characteristic curve [s] | REAL | 0.0 |
| ActRespUp2 | y value 2, actual open response time characteristic curve [s] | REAL | 1.0 |
| ActRespUp3 | y value 3, actual open response time characteristic curve [s] | REAL | 2.0 |
| ActRespUp4 | y value 4, actual open response time characteristic curve [s] | REAL | 3.0 |
| ActRespUp5 | y value 5, actual open response time characteristic curve [s] | REAL | 4.0 |
| ActRespUp6 | y value 6, actual open response time characteristic curve [s] | REAL | 5.0 |
| ActRespUp7 | y value 7, actual open response time characteristic curve [s] | REAL | 6.0 |
| ActRespUp8 | y value 8, actual open response time characteristic curve [s] | REAL | 7.0 |
| ActRespUp9 | y value 9, actual open response time characteristic curve [s] | REAL | 8.0 |
| ActRespUp10 | y value 10, actual open response time characteristic curve [s] | REAL | 9.0 |
| ActRespUp11 | y value 11, actual open response time characteristic curve [s] | REAL | 10.0 |
| ActRespUp12 | y value 12, actual open response time characteristic curve [s] | REAL | 11.0 |
| ActRespUp13 | y value 13, actual open response time characteristic curve [s] | REAL | 12.0 |
| ActRespUp14 | y value 14, actual open response time characteristic curve [s] | REAL | 13.0 |
| ActRespUp15 | y value 15, actual open response time characteristic curve [s] | REAL | 14.0 |
| ActRespUp16 | y value 16, actual open response time characteristic curve [s] | REAL | 15.0 |
| ActRespUp17 | y value 17, actual open response time characteristic curve [s] | REAL | 16.0 |
| ActRespUp18 | y value 18, actual open response time characteristic curve [s] | REAL | 17.0 |
| ActRespUp19 | y value 19, actual open response time characteristic curve [s] | REAL | 18.0 |
| ActRespUp20 | y value 20, actual open response time characteristic curve [s] | REAL | 19.0 |
| ActStepHiDn1 | x value 1, actual close response time characteristic curve | REAL | 5.0 |
| ActStepHiDn2 | x value 2, actual close response time characteristic curve | REAL | 10.0 |
| ActStepHiDn3 | x value 3, actual close response time characteristic curve | REAL | 15.0 |
| ActStepHiDn4 | x value 4, actual close response time characteristic curve | REAL | 20.0 |
| ActStepHiDn5 | x value 5, actual close response time characteristic curve | REAL | 25.0 |
| ActStepHiDn6 | x value 6, actual close response time characteristic curve | REAL | 30.0 |
| ActStepHiDn7 | x value 7, actual close response time characteristic curve | REAL | 35.0 |
| ActStepHiDn8 | x value 8, actual close response time characteristic curve | REAL | 40.0 |
| ActStepHiDn9 | x value 9, actual close response time characteristic curve | REAL | 45.0 |
| ActStepHiDn10 | x value 10, actual close response time characteristic curve | REAL | 50.0 |
| ActStepHiDn11 | x value 11, actual close response time characteristic curve | REAL | 55.0 |
| ActStepHiDn12 | x value 12, actual close response time characteristic curve | REAL | 60.0 |
| ActStepHiDn13 | x value 13, actual close response time characteristic curve | REAL | 65.0 |
| ActStepHiDn14 | x value 14, actual close response time characteristic curve | REAL | 70.0 |
| ActStepHiDn15 | x value 15, actual close response time characteristic curve | REAL | 75.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ActStepHiDn16 | x value 16, actual close response time characteristic curve | REAL | 80.0 |
| ActStepHiDn17 | x value 17, actual close response time characteristic curve | REAL | 85.0 |
| ActStepHiDn18 | x value 18, actual close response time characteristic curve | REAL | 90.0 |
| ActStepHiDn19 | x value 19, actual close response time characteristic curve | REAL | 95.0 |
| ActStepHiDn20 | x value 20, actual close response time characteristic curve | REAL | 100.0 |
| ActStepHiUp1 | x value 1, actual open response time characteristic curve | REAL | 5.0 |
| ActStepHiUp2 | x value 2, actual open response time characteristic curve | REAL | 10.0 |
| ActStepHiUp3 | x value 3, actual open response time characteristic curve | REAL | 15.0 |
| ActStepHiUp4 | x value 4, actual open response time characteristic curve | REAL | 20.0 |
| ActStepHiUp5 | x value 5, actual open response time characteristic curve | REAL | 25.0 |
| ActStepHiUp6 | x value 6, actual open response time characteristic curve | REAL | 30.0 |
| ActStepHiUp7 | x value 7, actual open response time characteristic curve | REAL | 35.0 |
| ActStepHiUp8 | x value 8, actual open response time characteristic curve | REAL | 40.0 |
| ActStepHiUp9 | x value 9, actual open response time characteristic curve | REAL | 45.0 |
| ActStepHiUp10 | x value 10, actual open response time characteristic curve | REAL | 50.0 |
| ActStepHiUp11 | x value 11, actual open response time characteristic curve | REAL | 55.0 |
| ActStepHiUp12 | x value 12, actual open response time characteristic curve | REAL | 60.0 |
| ActStepHiUp13 | x value 13, actual open response time characteristic curve | REAL | 65.0 |
| ActStepHiUp14 | x value 14, actual open response time characteristic curve | REAL | 70.0 |
| ActStepHiUp15 | x value 15, actual open response time characteristic curve | REAL | 75.0 |
| ActStepHiUp16 | x value 16, actual open response time characteristic curve | REAL | 80.0 |
| ActStepHiUp17 | x value 17, actual open response time characteristic curve | REAL | 85.0 |
| ActStepHiUp18 | x value 18, actual open response time characteristic curve | REAL | 90.0 |
| ActStepHiUp19 | x value 19, actual open response time characteristic curve | REAL | 95.0 |
| ActStepHiUp20 | x value 20, actual open response time characteristic curve | REAL | 100.0 |
| ActVlvPos1 | x value 1, actual flow characteristic curve | REAL | 0.0 |
| ActVlvPos2 | x value 2, actual flow characteristic curve | REAL | 5.0 |
| ActVlvPos3 | x value 3, actual flow characteristic curve | REAL | 10.0 |
| ActVlvPos4 | x value 4, actual flow characteristic curve | REAL | 20.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ActVlvPos5 | x value 5, actual flow characteristic curve | REAL | 25.0 |
| ActVlvPos6 | x value 6, actual flow characteristic curve | REAL | 30.0 |
| ActVlvPos7 | x value 7, actual flow characteristic curve | REAL | 35.0 |
| ActVlvPos8 | x value 8, actual flow characteristic curve | REAL | 40.0 |
| ActVlvPos9 | x value 9, actual flow characteristic curve | REAL | 45.0 |
| ActVlvPos10 | x value 10, actual flow characteristic curve | REAL | 50.0 |
| ActVlvPos11 | x value 11, actual flow characteristic curve | REAL | 55.0 |
| ActVlvPos12 | x value 12, actual flow characteristic curve | REAL | 60.0 |
| ActVlvPos13 | x value 13, actual flow characteristic curve | REAL | 65.0 |
| ActVlvPos14 | x value 14, actual flow characteristic curve | REAL | 70.0 |
| ActVlvPos15 | x value 15, actual flow characteristic curve | REAL | 75.0 |
| ActVlvPos16 | x value 16, actual flow characteristic curve | REAL | 80.0 |
| ActVlvPos17 | x value 17, actual flow characteristic curve | REAL | 85.0 |
| ActVlvPos18 | x value 18, actual flow characteristic curve | REAL | 90.0 |
| ActVlvPos19 | x value 19, actual flow characteristic curve | REAL | 95.0 |
| ActVlvPos20 | x value 20, actual flow characteristic curve | REAL | 100.0 |
| CntHisStepDn | Indicates the number of negative steps | DINT | 0 |
| CntHisStepUp | Indicates the number of positive steps | DINT | 0 |
| CtrlDevMaxAct | 1 = Maximum deviation between end position reached and setpoint exceeded | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DeadAct | 1 = Dead time monitoring active (available in a later version) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DeadTolOut | Tolerance for dead time monitoring (available in a later version) | REAL | 0.0 |
| DevFlow | Deviation from the set flow characteristic curve | REAL | 0.0 |
| DevRespDn | Deviation from the set close response time characteristic curve | REAL | 0.0 |
| DevRespUp | Deviation from the set open response time characteristic curve | REAL | 0.0 |
| ER | Manipulated variable difference | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ErrorNum | Output of the existing error number. For error numbers that can be output by this block, see VlvMon error handling (Page 581). | INT | -1 |
| ExtMoveAct | 1 = External movement detected | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkCloseOut | 1 = Control valve is closed | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| FbkClsgOut | Control valve closing feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkOpenOut | 1 = Control valve is open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkOpngOut | Control valve opening feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FinalPos | End position reached after a valve movement | REAL | 0.0 |
| FlowAct | 1 = Deviation from the set flow characteristic curve is too large | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FlowDriftAct | 1 = Deviation between set flow and actual flow characteristic curve is too large | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| HisPos0 | Indicates the time in closed status [%] (with reference to T_HisPos) | REAL | 0.0 |
| HisPos1 | Indicates the duration of stay in the valve position range 0-10% [%] (with reference to T_HisPos) | REAL | 0.0 |
| HisPos2 | Indicates the duration of stay in the valve position range 10-20% [%] (with reference to T_HisPos) | REAL | 0.0 |
| HisPos3 | Indicates the duration of stay in the valve position range 20-30% [%] (with reference to T_HisPos) | REAL | 0.0 |
| HisPos4 | Indicates the duration of stay in the valve position range 30-40% [%] (with reference to T_HisPos) | REAL | 0.0 |
| HisPos5 | Indicates the duration of stay in the valve position range 40-50% [%] (with reference to T_HisPos) | REAL | 0.0 |
| HisPos6 | Indicates the duration of stay in the valve position range 50-60% [%] (with reference to T_HisPos) | REAL | 0.0 |
| HisPos7 | Indicates the duration of stay in the valve position range 60-70% [%] (with reference to T_HisPos) | REAL | 0.0 |
| HisPos8 | Indicates the duration of stay in the valve position range 70-80% [%] (with reference to T_HisPos) | REAL | 0.0 |
| HisPos9 | Indicates the duration of stay in the valve position range 80-90% [%] (with reference to T_HisPos) | REAL | 0.0 |
| HisPos10 | Indicates the duration of stay in the valve position range 90-100% [%] (with reference to T_HisPos) | REAL | 0.0 |
| HisStepHiDn1 | Indicates the number of negative steps in the step change range 0-10% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiDn2 | Indicates the number of negative steps in the step change range 10-20% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiDn3 | Indicates the number of negative steps in the step change range 20-30% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiDn4 | Indicates the number of negative steps in the step change range 30-40% (with reference to CntHisStepDn) | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| HisStepHiDn5 | Indicates the number of negative steps in the step change range 40-50% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiDn6 | Indicates the number of negative steps in the step change range 50-60% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiDn7 | Indicates the number of negative steps in the step change range 60-70% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiDn8 | Indicates the number of negative steps in the step change range 70-80% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiDn9 | Indicates the number of negative steps in the step change range 80-90% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiDn10 | Indicates the number of negative steps in the step change range 90-100% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiUp1 | Indicates the number of positive steps in the step change range 0-10% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiUp2 | Indicates the number of positive steps in the step change range 10-20% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiUp3 | Indicates the number of positive steps in the step change range 20-30% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiUp4 | Indicates the number of positive steps in the step change range 30-40% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiUp5 | Indicates the number of positive steps in the step change range 40-50% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiUp6 | Indicates the number of positive steps in the step change range 50-60% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiUp7 | Indicates the number of positive steps in the step change range 60-70% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiUp8 | Indicates the number of positive steps in the step change range 70-80% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiUp9 | Indicates the number of positive steps in the step change range 80-90% (with reference to CntHisStepDn) | REAL | 0.0 |
| HisStepHiUp10 | Indicates the number of positive steps in the step change range 90-100% (with reference to CntHisStepDn) | REAL | 0.0 |
| OnAct | 1 = Block is "On" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS1PermLog | Display of OS1Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1Perm | DWORD | 16#FFFFFFFF |
| PosReached | 1 = Control valve has reached specified position | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RelFlow | Relative deviation from the set flow characteristic curve | REAL | 0.0 |
| RelFlowDrift | Relative deviation between set flow and actual flow characteristic curve | REAL | 0.0 |
| RelRespDn | Relative deviation from the set close response time characteristic curve | REAL | 0.0 |
| RelRespUp | Relative deviation from the set open response time characteristic curve | REAL | 0.0 |
| RespDnAct | 1 = Deviation from the set close response time characteristic curve is too large | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RespUpAct | 1 = Deviation from the set open response time characteristic curve is too large | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SID | Indicates the number of direction changes | | |
| SID_MaxAct | 1 = Maximum number of direction changes exceeded | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SID_WearRes | Indicates the wear reserve for the number of direction changes | REAL | 0.0 |
| siFlow | Indicates the normalized flow [$m^3/h$] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| siFlowNorm | Indicates the pressure-normalized flow | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| siP_In | Indicates the normalized input pressure [bar] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| siP_Out | Indicates the normalized output pressure [bar] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 | DWORD | 16#0 |
| Status2 | Status word 2 | DWORD | 16#0 |
| StepHiDn | Indicates the last detected negative step change | REAL | 0.0 |
| StepHiUp | Indicates the last detected positive step change | REAL | 0.0 |
| Strokes | Indicates the number of complete valve strokes | DINT | 0 |
| StrokesMaxAct | 1 = Maximum number of valve strokes exceeded | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StrokesWearRes | Indicates the wear reserve for the number of completed valve strokes | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SuppPresAct | 1 = Supply pressure error | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| T_HisPos | Indicates the monitoring time for the valve position histogram | REAL | 0.0 |
| T_Idle | Indicates the downtime since the last detected movement | DINT | 0 |
| T_IdleMaxAct | 1 = Maximum downtime exceeded | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| T_Move | Indicates the time in motion | DINT | 0 |
| T_MoveMaxAct | 1 = Maximum in continuous motion exceeded | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| T_RespDn | Indicates the response time of the last detected negative step | REAL | 0.0 |
| T_RespUp | Indicates the response time of the last detected positive step | REAL | 0.0 |
| VlvPosMaxAct | 1 = Upper valve stop exceeded | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| VlvPosMinAct | 1 = Lower valve stop undershot | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| WearReserveFlowDrift | Wear reserve for set/actual difference of flow characteristic curve | REAL | 0.0 |

## 4.3.8 VlvMon block diagram

A block diagram is not provided for this block.

## 4.3.9 VlvMon additional information

### Relationship between measuring variables and diagnosable problems

| | Flow | Pressure difference across the valve | Valve set position | Valve actual position |
|---|---|---|---|---|
| Time without movement | | | x | x |
| Time without standstill | | | x | x |
| Number of strokes | | | x | x |
| Directional reversal | | | x | x |
| External movement | | | x | x |
| Violation of high/low end stop | | | | x |

| | Flow | Pressure difference across the valve | Valve set position | Valve actual position |
|---|---|---|---|---|
| Reaction time characteristic curves | | | x | x |
| Deviation | | | x | x |
| Flow characteristic curve | x | x | | x |
| Histogram valve position | | | x | x |
| Histogram step changes | | | x | x |

Annotations:

x: Measured value required

## 4.3.10 Operator control and monitoring

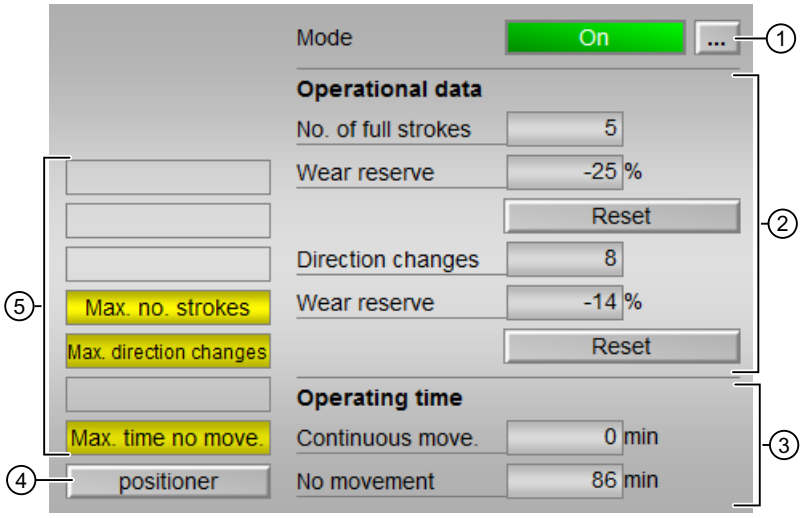
### 4.3.10.1 VlvMon views

The VlvMon block provides the following views:

- Standard view of VlvMon (Page 598)
- Flow characteristic curve (Page 599)
- Reaction time characteristic curves (Page 601)
- Histogram (Page 604)
- Limit values (Page 606)
- Parameter view 1 (Page 608)
- Parameter view 2 (Page 610)
- Memo view (Page 338)
- Block icon for PumpMon (Page 612)

For general information on the faceplate and block icon, refer to the Structure of the faceplate (Page 279) and Block icon structure (Page 260).

### 4.3.10.2 VlvMon standard view




**(1) Display and switchover of operating mode**
This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On
- Out of service

**(2) Operating data**
This area shows the current number of full valve strokes (`Strokes`) and changes of direction (`SID`). In addition, the percentage of reserve for the maximum value (`StrokesWearRes`, `SID_WearRes`) is specified for the stroke and change of direction.

A reset button is provided in each case to reset the strokes (`RstStrokesOp`) and change of direction (`RstSID_Op`).

**(3) Operating time**

This area shows the currently calculated time of continuous movement (`T_Move`) and the continuous idle time (`T_Idle`). The values cannot be operated.

**(4) Navigation button for switching to the standard view of any faceplate**

This display is only visible when the corresponding `MV` block input is connected. Use this jump button to reach the standard view of the interconnected block.

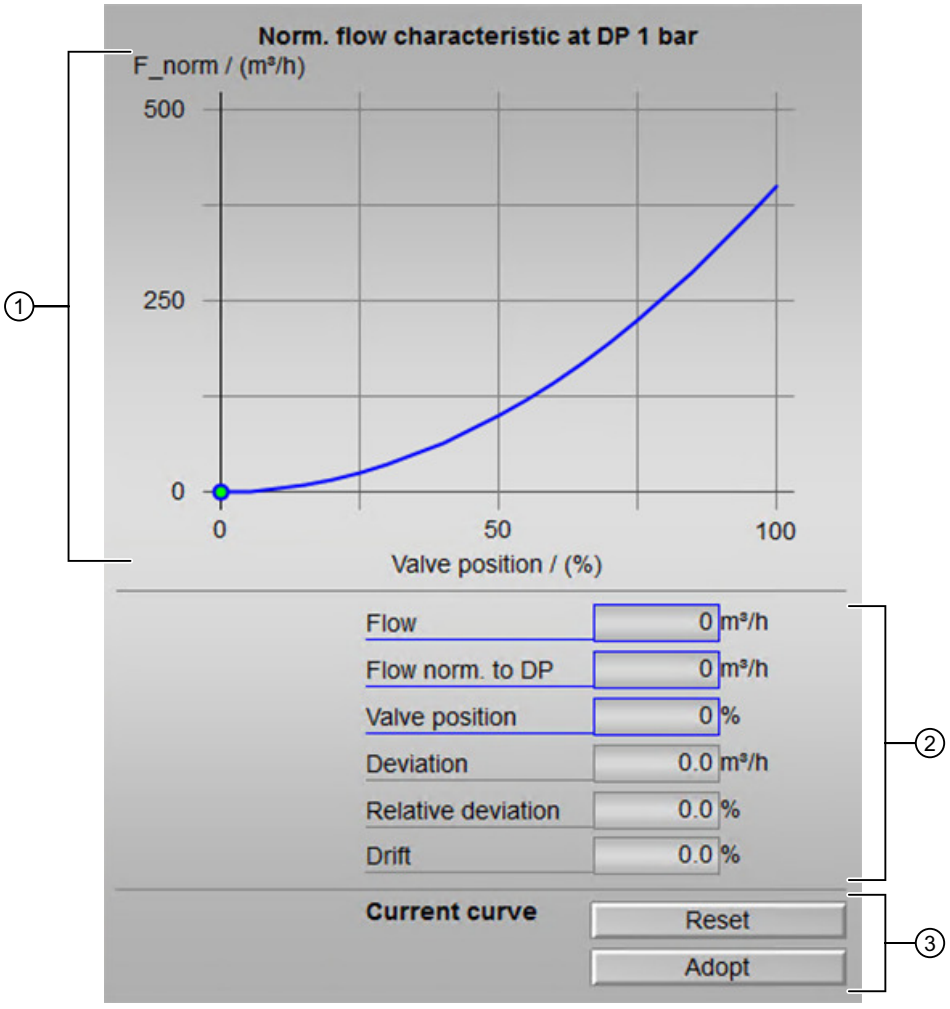
**(5) Display area for block states**

This area provides additional information on the operating state of the block:

- External movement
- Deviation
- Supply pressure
- Max. number of strokes
- Max. direction changes

- End stop high / end stop low
- Max. idle time / max. movement time

### 4.3.10.3 VlvMon flow characteristic curve view




**(1) Display of flow characteristic curve with automatically detected actual characteristic curve**

The interpolation points for the set characteristic curve (solid blue line) are entered in the "Parameter 1" view.

The operating point continually shows the current standardized flow (`siFlow.Value`) for the current valve position (`Rbk.Value`).

If the distance between the current operating point and the set characteristic curve exceeds a specified tolerance (`FlowTol`), the color of the operating point changes from green to red and the corresponding output parameter (`FlowAct.Value`) is activated.

In addition to the set characteristic curve, the automatically detected actual characteristic curve is shown (dashed cyan line). The up-to-dateness of the characteristic curve is indicated by black dots. The larger a dot is shown, the more recent are the associated measured values. If a valve

position has not yet been approached and there is therefore no measured value for the corresponding interpolation point of the characteristic curve, no dot is shown.

The Y axis scaling is dynamic:

- Y: 0 -> `FlowMax` / 2-> `FlowMax`

The scaling of the X axis is static:

- X: 0 -> 50 % -> 100 %

**Note**

If `Feature.Bit10` "Flow characteristic curve active" = `FALSE`, the characteristic curves and the operating point are hidden.

**(2) Display of the characteristic values**

This area shows the following values:

- Flow (`Flow.Value`)
- Standardized flow (`siFlow.Value`)
- Valve position (`Rbk.Value`)
- Absolute deviation of the operating point from the set characteristic curve (`DevFlow`)
- Relative deviation of the operating point from the set characteristic curve (`RelFlow`)
- Relative deviation of the actual characteristic curve from the set characteristic curve (`RelFlowDrift`)

The values cannot be operated.

**Note**

No unit is displayed in the faceplate for the normalized flow. The unit of computation however would be $\frac{m^3/h}{\sqrt{bar}}$

**(3) Function for actual characteristic**
This area provides a button for adopting and a button for resetting the actual characteristic curve.

When the actual characteristic curve is adopted (`Adopt_ActFlow = 1`), the actual characteristic curve is written to the corresponding set characteristic curve.

- `Flow1 … Flow20 = ActFlow1 … ActFlow20`
- `VlvPos1 … VlvPos20 = ActVlvPos1 … ActVlvPos20`

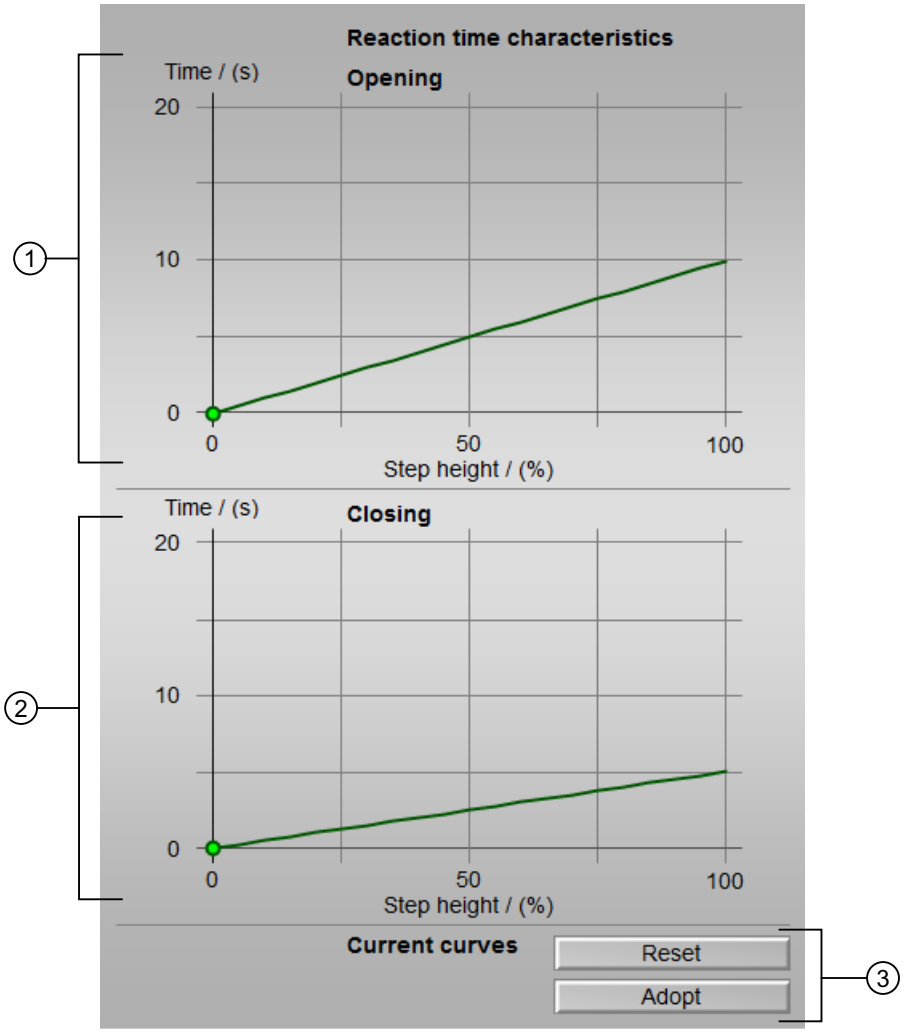
When the actual characteristic curve is reset (`Rst_ActFlow = 1`), the actual characteristic curve and the associated points for displaying the up-to-dateness are reset to zero.

- `ActFlow1 … ActFlow20= 0.0`
- `ActVlvPos1 … ActVlvPos20= 0.0`
- `AccFlowPoint1 … AccFlowPoint20 = 0.0`

**Note**

If `Feature.Bit10` "Flow characteristic curve active" = `FALSE`, the buttons for adopting and resetting the actual characteristic curve are disabled.

### 4.3.10.4 VlvMon reaction time characteristic curve view




**(1) Display of the response time characteristic with automatically detected "Open" actual characteristic**
The interpolation points for the set characteristic curve (solid dark green) are entered in the "Parameter 2" view.

The operating point shows the last detected positive setpoint change (`StepHiUp`, `T_RespUp`). It remains unchanged until a new step has been detected.

If the distance between the current operating point and the set characteristic curve exceeds a specified tolerance (`RespTol`), the color of the operating point changes from green to red and the corresponding output parameter (`RespUpAct.Value`) is activated.

In addition to the set characteristic curve, the automatically detected actual characteristic curve is shown (dashed green line).

The Y axis scaling is dynamic:

- Y: 0 -> `T_RespMax` / 2 -> `T_RespMax`

The scaling of the X axis is static:

- X: 0 -> 50 % -> 100 %

**Note**

If `Feature` Bit 11 "Response time characteristic curve "Open" active" = `FALSE`, the characteristic curves and the operating point are hidden.

**(2) Display of response time characteristic curve with automatically detected actual "Close" characteristic curve**

The interpolation points for the set characteristic curve (solid dark green) are entered in the "Parameter 2" view.

The operating point shows the last detected negative setpoint change (`StepHiDn`, `T_RespDn`). It remains unchanged until a new step has been detected.

If the distance between the current operating point and the set characteristic curve exceeds a specified tolerance (`RespTol`), the color of the operating point changes from green to red and the corresponding output parameter (`RespDnAct.Value`) is activated.

In addition to the set characteristic curve, the automatically detected actual characteristic curve is shown (dashed green line).

The Y axis scaling is dynamic:

- Y: 0 -> `T_RespMax` / 2 -> `T_RespMax`

The scaling of the X axis is static:

- X: 0 -> 50 % -> 100 %

**Note**

If `Feature` Bit 12 "Response time characteristic curve "Close" active" = `FALSE`, no characteristic curves or operating point are displayed.

**(3) Functions for the actual characteristic curve**

This area provides a button for adopting and a button for resetting the actual characteristic curves.

When the actual characteristic curve is adopted (`Adopt_ActResp = 1`), the two actual characteristic curves are written to the corresponding set characteristic curves.

- `StepHiUp1 … StepHiUp20= ActStepHiUp1 … ActStepHiUp20`
- `RespUp1 … RespUp20= ActRespUp1 … ActRespUp20`
- `StepHiDn1 … StepHiDn20 = ActStepHiDn1 … ActStepHiDn20`
- `RespDn1 … RespDn20= ActRespDn1 … ActRespDn20`

When the actual characteristic curve is reset (`Rst_ActResp = 1`), the two actual characteristic curves are reset to zero

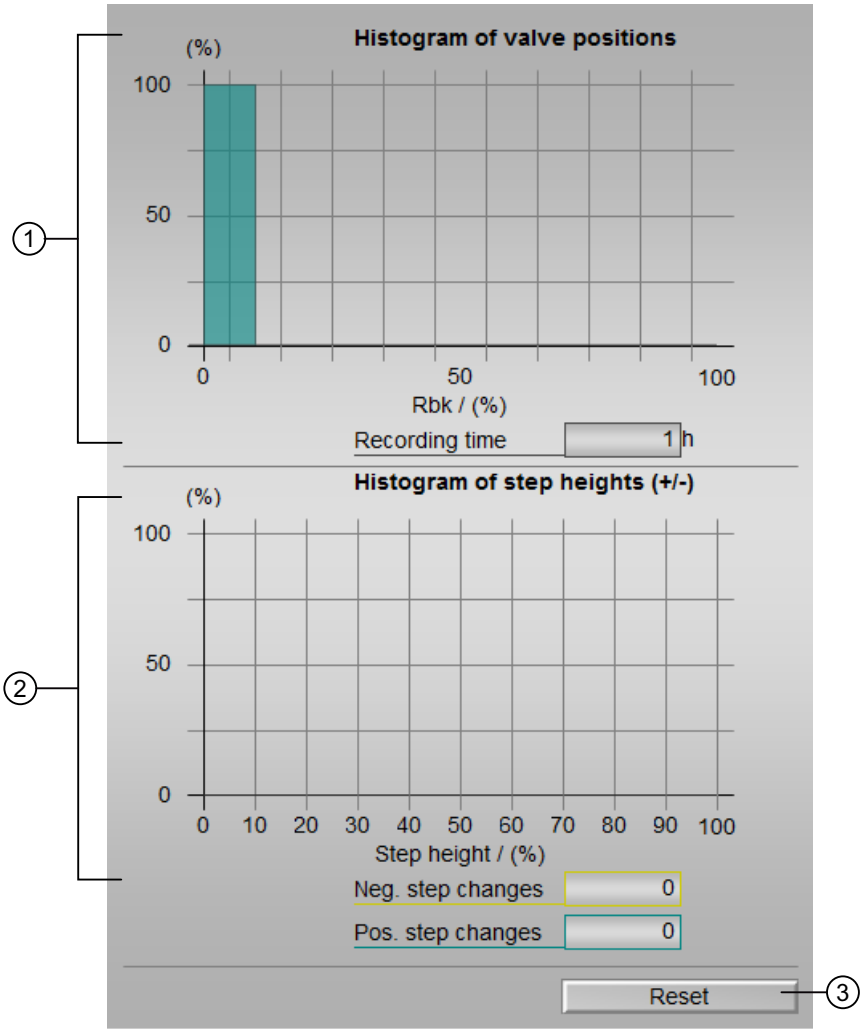
- `ActStepHiUp1 … ActStepHiUp20= 0.0`
- `ActRespUp1 … ActRespUp20= 0.0`
- `ActStepHiDn1 … ActStepHiDn20= 0.0`
- `ActRespDn1 … ActRespDn20= 0.0`

and the operating points are shifted to the point of origin.

**Note**

If `Feature` Bits 11 and 12 = `FALSE`, the buttons for adopting and resetting the actual characteristic curves are disabled.

### 4.3.10.5 VlvMon histogram view




The histograms show the frequency distributions of the various valve positions or setpoint step changes over the entire travel range (0 to 100%) of the valve position as a bar graph.

**(1) Histogram for valve position**

The histogram consists of a total of 11 bars. The first bar on the left of the Y axis (purple) designates the time in the closed state (`HisPos0`). The other 10 bars (cyan) divide the valve position range `0 ... 100%` into 10 equal sub-ranges. Each of these sub-ranges is assigned a counter for the duration of stay. The following rules apply:

- `HisPos1` corresponds to >0-10% valve position
- `HisPos2` corresponds to 10-20% valve position
- `HisPos3` corresponds to 20-30% valve position
- `HisPos4` corresponds to 30-40% valve position
- `HisPos5` corresponds to 40-50% valve position
- `HisPos6` corresponds to 50-60% valve position

- HisPos7 corresponds to 60-70% valve position
- HisPos8 corresponds to 70-80% valve position
- HisPos9 corresponds to 80-90% valve position
- HisPos10 corresponds to 90-100% valve position

The following applies to the counter which indicates the "closed" state:

- HisPos0 corresponds to the "closed" state (FbkCloseOut.Value = TRUE)

The scaling of the X and Y axis is static:

- X: 0 -> 50% -> 100%
- Y: 0 -> 50% -> 100%

The "Recording period:" field shows the period in [h] (T_HisPos) during which these statistics were gathered.

**(2) Histogram for step changes (+/-)**

This histogram consists of a total of 20 bars, 10 bars for positive steps (turquoise) and 10 bars for negative steps (yellow). The bars provide an overview in percentage of distribution in relation to the total number of positive and negative steps.

The valve position range 0...100% is divided into 10 equal sub-ranges. A bar is allocated to each of these sub-ranges for positive and negative steps in this range. The following applies to positive steps:

- HisStepHiUp1 corresponds to step change 0-10%
- HisStepHiUp2 corresponds to step change 10-20%
- HisStepHiUp3 corresponds to step change 20-30%
- HisStepHiUp4 corresponds to step change 30-40%
- HisStepHiUp5 corresponds to step change 40-50%
- HisStepHiUp6 corresponds to step change 50-60%
- HisStepHiUp7 corresponds to step change 60-70%
- HisStepHiUp8 corresponds to step change 70-80%
- HisStepHiUp9 corresponds to step change 80-90%
- HisStepHiUp10 corresponds to step change 90-100%

The following applies to negative steps:

- HisStepHiDn1 corresponds to step change 0 to -10%
- HisStepHiDn2 corresponds to step change -10 to -20%
- HisStepHiDn3 corresponds to step change -20 to -30%
- HisStepHiDn4 corresponds to step change -30 to -40%
- HisStepHiDn5 corresponds to step change -40 to -50%
- HisStepHiDn6 corresponds to step change -50 to -60%
- HisStepHiDn7 corresponds to step change -60 to -70%
- HisStepHiDn8 corresponds to step change -70 to -80%

- `HisStepHiDn9` corresponds to step change -80 to -90%
- `HisStepHiDn10` corresponds to step change -90 to -100%

The scaling of the X and Y axis is static:

- X: 0 -> 50% -> 100%
- Y: 0 -> 50% -> 100%

The "No. pos. steps:" field shows the total number (`CntHisStepUp`) of positive steps.

The "No. pos. steps:" field shows the total number (`CntHisStepDN`) of negative steps.

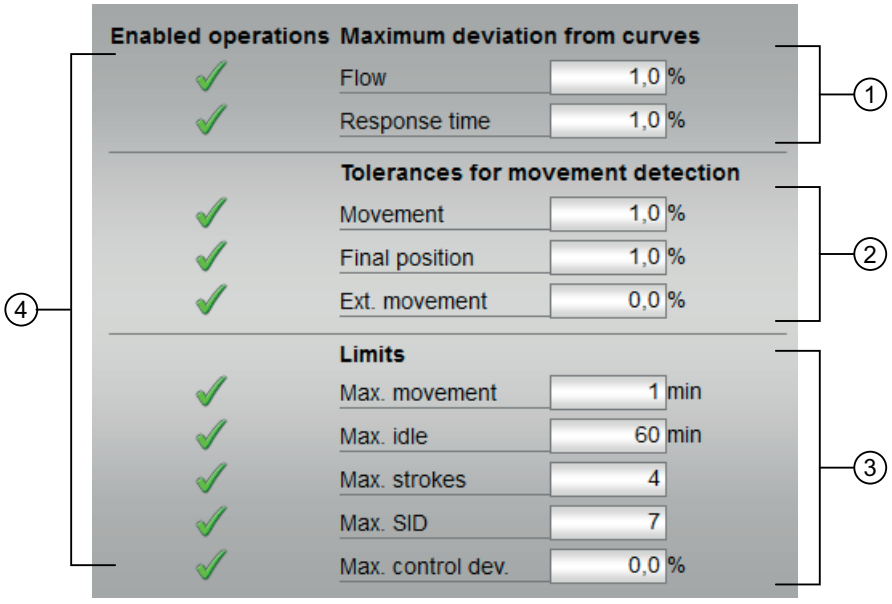
**(3) Resetting the histogram data**

The reset button can be used to reset all the stored values to zero and restart.

**Note**

All collected statistics will be lost with this action!

### 4.3.10.6 VlvMon limit view




**(1) Maximum deviation from the characteristic curves**

In this area, you can enter the tolerances for the deviation from the characteristic curves.

You can change the following tolerances:

- "Flow:" Maximum deviation from the flow characteristic curve (`FlowTol`)
- "Response time": Maximum deviation from the reaction time characteristic curve (`RespTol`)

**(2) Tolerances for motion detection**

The following tolerances can be entered in this area:

- "Movement:" Tolerance for the detection of a valve movement (`MoveTol`)
- "Set position:" Tolerance for detection of the set position being reached (`FinalTol`)
- "External movement:" Tolerance for the detection of an external movement without setpoint change (`ExtMoveTol`)

**(3) Limits**

The following limits can be entered in this area:

- "Max. movement": Limit for the maximum movement time without standstill (`T_MoveMax`)
- "Max. idle": Limit for the maximum idle time without movement (`T_IdleMax`)
- "Max. number of strokes": Limit for the maximum number of full (100%) valve strokes (`StrokesMax`)
- "Max. direction changes": Limit for the maximum number of changes of direction (`SID_Max`)
- "Max. control deviation": Limit for the maximum control deviation after reaching the set position (`CtrlDevMax`)

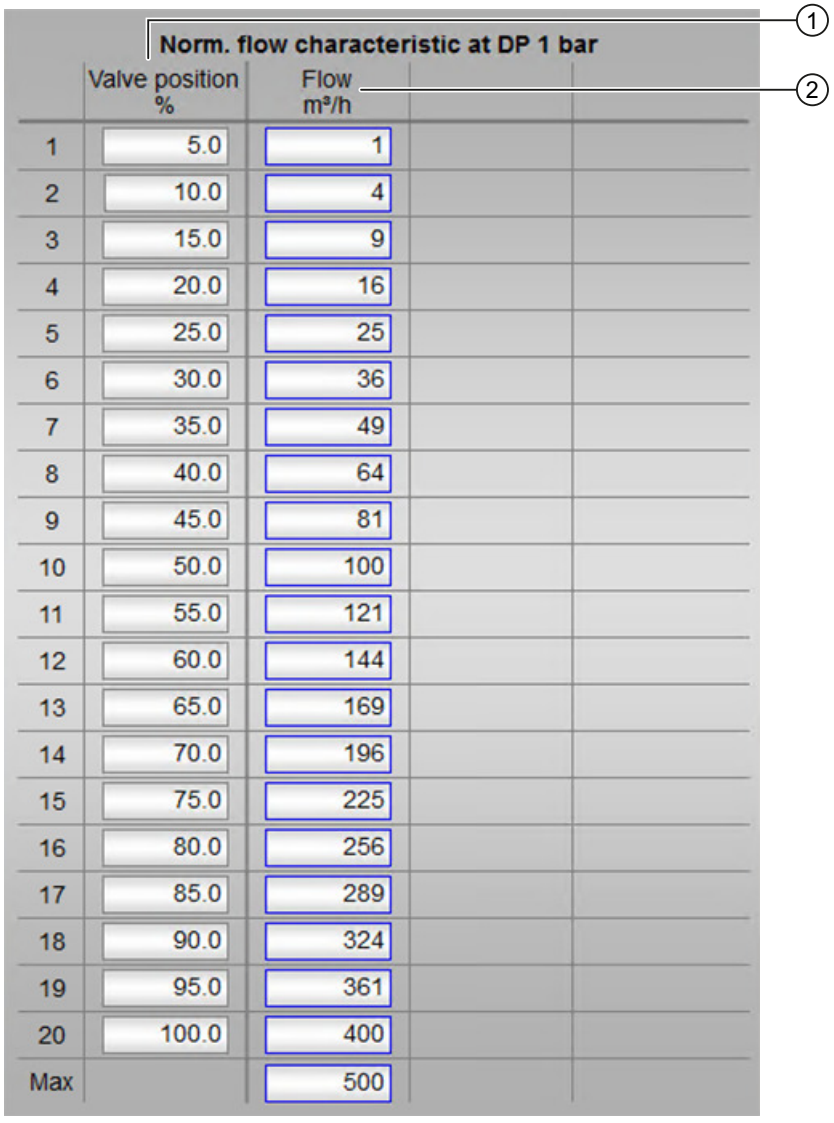
**(4) Enabled operations**

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark:** The OS operator can control this parameter
- **Gray check mark:** Access to this parameter is temporarily disabled for the OS operator due to process conditions
- **Red cross:** The OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

### 4.3.10.7 VlvMon parameter 1 view




| | Valve position % | Flow m³/h |
|---|---|---|
| 1 | 5.0 | 1 |
| 2 | 10.0 | 4 |
| 3 | 15.0 | 9 |
| 4 | 20.0 | 16 |
| 5 | 25.0 | 25 |
| 6 | 30.0 | 36 |
| 7 | 35.0 | 49 |
| 8 | 40.0 | 64 |
| 9 | 45.0 | 81 |
| 10 | 50.0 | 100 |
| 11 | 55.0 | 121 |
| 12 | 60.0 | 144 |
| 13 | 65.0 | 169 |
| 14 | 70.0 | 196 |
| 15 | 75.0 | 225 |
| 16 | 80.0 | 256 |
| 17 | 85.0 | 289 |
| 18 | 90.0 | 324 |
| 19 | 95.0 | 361 |
| 20 | 100.0 | 400 |
| Max | | 500 |

The interpolation points for the characteristic curve can be read from the data sheet and entered here.

The fields for the characteristic curve interpolation points are enabled based on whether `Feature` Bit 10 "Flow characteristic curve active" is set.

**(1) Valve position parameters**

The 20 interpolation points of the valve positions provide the X axis for the flow characteristic curves. Here, each interpolation point is assigned to a valve position range. The following rules apply:

- `VlvPos1` corresponds to 0-5% valve position
- `VlvPos2` corresponds to 5-10% valve position
- `VlvPos3` corresponds to 10-15% valve position

- VlvPos4 corresponds to 15-20% valve position
- VlvPos5 corresponds to 20-25% valve position
- VlvPos6 corresponds to 25-30% valve position
- VlvPos7 corresponds to 30-35% valve position
- VlvPos8 corresponds to 35-40% valve position
- VlvPos9 corresponds to 40-45% valve position
- VlvPos10 corresponds to 45-50% valve position
- VlvPos11 corresponds to 50-55% valve position
- VlvPos12 corresponds to 55-60% valve position
- VlvPos13 corresponds to 60-65% valve position
- VlvPos14 corresponds to 65-70% valve position
- VlvPos15 corresponds to 70-75% valve position
- VlvPos16 corresponds to 75-80% valve position
- VlvPos17 corresponds to 80-85% valve position
- VlvPos18 corresponds to 85-90% valve position
- VlvPos19 corresponds to 90-95% valve position
- VlvPos20 corresponds to 95-100% valve position

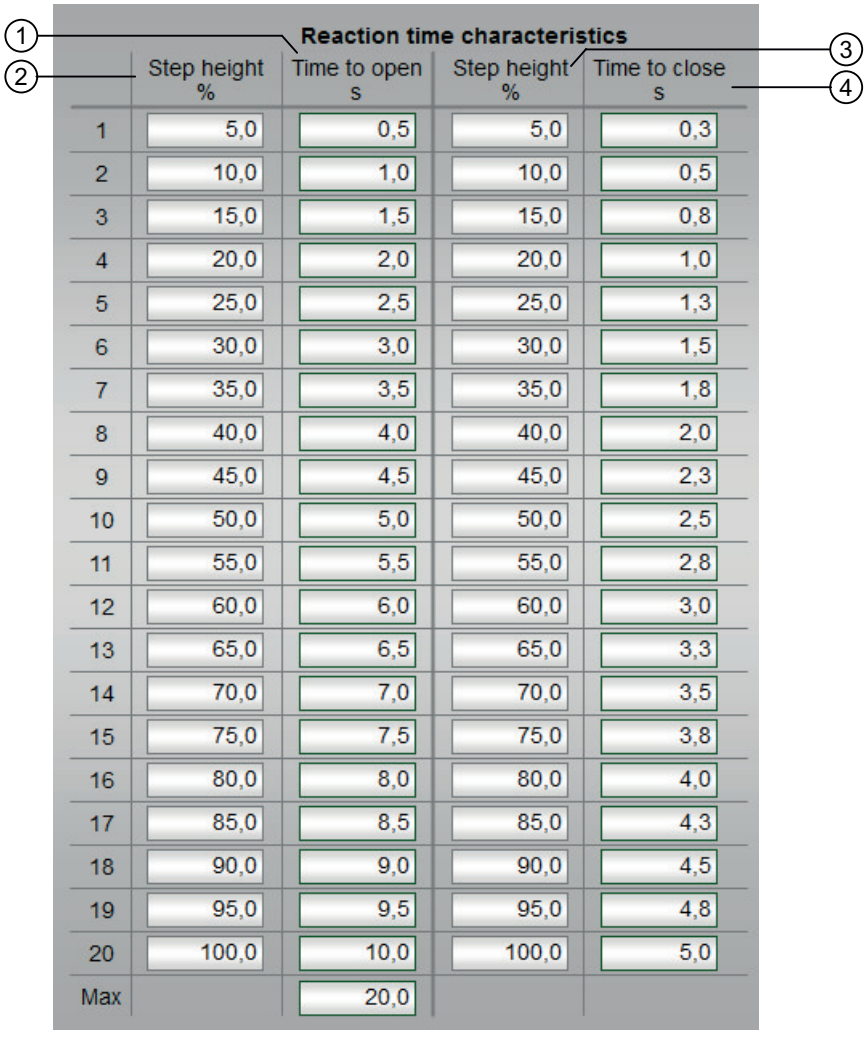
The display is within the range:

```
0 <= VlvPos1 <= ... <= VlvPos20 <= 100%
```

**(2) Flow parameters**

The flow parameters Flow1 to Flow20, as well as the maximum value FlowMax, are specified in this area. Parameters that have the value 0 are ignored in the characteristic curve.

### 4.3.10.8 VlvMon parameter 2 view



| | Reaction time characteristics | | | |
|---|---|---|---|---|
| | Step height % | Time to open s | Step height % | Time to close s |
| 1 | 5,0 | 0,5 | 5,0 | 0,3 |
| 2 | 10,0 | 1,0 | 10,0 | 0,5 |
| 3 | 15,0 | 1,5 | 15,0 | 0,8 |
| 4 | 20,0 | 2,0 | 20,0 | 1,0 |
| 5 | 25,0 | 2,5 | 25,0 | 1,3 |
| 6 | 30,0 | 3,0 | 30,0 | 1,5 |
| 7 | 35,0 | 3,5 | 35,0 | 1,8 |
| 8 | 40,0 | 4,0 | 40,0 | 2,0 |
| 9 | 45,0 | 4,5 | 45,0 | 2,3 |
| 10 | 50,0 | 5,0 | 50,0 | 2,5 |
| 11 | 55,0 | 5,5 | 55,0 | 2,8 |
| 12 | 60,0 | 6,0 | 60,0 | 3,0 |
| 13 | 65,0 | 6,5 | 65,0 | 3,3 |
| 14 | 70,0 | 7,0 | 70,0 | 3,5 |
| 15 | 75,0 | 7,5 | 75,0 | 3,8 |
| 16 | 80,0 | 8,0 | 80,0 | 4,0 |
| 17 | 85,0 | 8,5 | 85,0 | 4,3 |
| 18 | 90,0 | 9,0 | 90,0 | 4,5 |
| 19 | 95,0 | 9,5 | 95,0 | 4,8 |
| 20 | 100,0 | 10,0 | 100,0 | 5,0 |
| Max | | 20,0 | | |

The interpolation points for the characteristic curves can be read from the data sheet and entered here.

The fields for the characteristic curve interpolation points (1 and 2) are enabled based on whether `Feature.Bit11` "Response time characteristic curve "Open" active" is set.

The fields for the characteristic curve interpolation points (3 and 4) are enabled based on whether `Feature.Bit12` "Response time characteristic curve "Close" active" is set.

**(1) Time UP**

The parameters `RespUp1` to `RespUp20`, as well as the maximum value `T_RespMax`, are specified in this area. Parameters that have the value 0 are ignored in the characteristic curve.

**(2) Step change UP**

The 20 interpolation points of the step change specify the x-axis for the response time characteristic curve "Open".

Here, each interpolation point is assigned to a step change range. The following rules apply:

- `StepHiUp1` corresponds to 0-5% valve step
- `StepHiUp2` corresponds to 5-10% valve step
- `StepHiUp3` corresponds to 10-15% valve step
- `StepHiUp4` corresponds to 15-20% valve step
- `StepHiUp5` corresponds to 20-25% valve step
- `StepHiUp6` corresponds to 25-30% valve step
- `StepHiUp7` corresponds to 30-35% valve step
- `StepHiUp8` corresponds to 35-40% valve step
- `StepHiUp9` corresponds to 40-45% valve step
- `StepHiUp10` corresponds to 45-50% valve step
- `StepHiUp11` corresponds to 50-55% valve step
- `StepHiUp12` corresponds to 55-60% valve step
- `StepHiUp13` corresponds to 60-65% valve step
- `StepHiUp14` corresponds to 65-70% valve step
- `StepHiUp15` corresponds to 70-75% valve step
- `StepHiUp16` corresponds to 75-80% valve step
- `StepHiUp17` corresponds to 80-85% valve step
- `StepHiUp18` corresponds to 85-90% valve step
- `StepHiUp19` corresponds to 90-95% valve step
- `StepHiUp20` corresponds to 95-100% valve step

The display is within the range:

```
0 <= StepHiUp1 <= ... <= StepHiUp20 <= 100%
```

**(3) Step change DOWN**

The 20 interpolation points of the step change provide the x-axis for the response time characteristic curve "Close".

Here, each interpolation point is assigned to a step change range. The following rules apply:

- `StepHiDn1` corresponds to 0 to -5% valve step
- `StepHiDn2` corresponds to -5 to -10% valve step
- `StepHiDn3` corresponds to -10 to -15% valve step
- `StepHiDn4` corresponds to -15 to -20% valve step
- `StepHiDn5` corresponds to -20 to -25% valve step
- `StepHiDn6` corresponds to -25 to -30% valve step
- `StepHiDn7` corresponds to -30 to -35% valve step
- `StepHiDn8` corresponds to -35 to -40% valve step

- StepHiDn9 corresponds to -40 to -45% valve step
- StepHiDn10 corresponds to -45 to -50% valve step
- StepHiDn11 corresponds to -50 to -55% valve step
- StepHiDn12 corresponds to -55 to -60% valve step
- StepHiDn13 corresponds to -60 to -65% valve step
- StepHiDn14 corresponds to -65 to -70% valve step
- StepHiDn15 corresponds to -70 to -75% valve step
- StepHiDn16 corresponds to -75 to -80% valve step
- StepHiDn17 corresponds to -80 to -85% valve step
- StepHiDn18 corresponds to -85 to -90% valve step
- StepHiDn19 corresponds to -90 to -95% valve step
- StepHiDn20 corresponds to -95 to -100% valve step

The display is within the range:

```
0 <= StepHiDn1 <= ... <= StepHiDn20 <= 100%
```

**(4) Time DOWN**

The parameters RespDn1 to RespDn20 are specified in this area. Parameters that have the value 0 are ignored in the characteristic curve.

**Note**

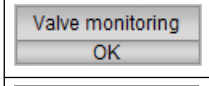
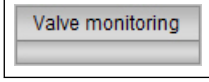
The maximum value of the response time characteristic curve "Open" also applies for this characteristic curve.

### 4.3.10.9 Block icon for VlvMon

A block icon with the following displays is available to you:

- Block type or process tag
- Operating modes
- Status of the valve from the view of the valve monitoring.

The block icon from template @TemplateAPLV9.PDL:

| Symbols | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in full display in the normal state |
|  | 1 | Block icon in full display when at least one error output is active |
|  | | Block icon in the "Out of service" operating mode |

In a control panel for the plant operator, it may be useful to make valve monitoring available via a shortcut from the valve faceplate, and place the block icon for the valve monitoring in only one maintenance view of the plant.

You can find additional information about the block icon and the operator control options in the block icon in the following sections:

- "Configuring the block icons (Page 267)"
- "Block icon structure (Page 260)"
- "Operation via the block icon (Page 269)".

# 4.4 Stationary processes - SteadyState

## 4.4.1 Description of SteadyState

### Object name (type + number) and family

Type + number: FB 1955

Family: CondMon

Number of process objects: 2 PO

### Area of application

The detection of steady states of a dynamic process and the steadiness of a signal is a useful aid for the open-loop control, closed-loop control and management of processes. There are often problems which make this detection difficult. The SteadyState function block is a solution which is suitable for many such problems.

The SteadyState function block analyzes an input signal and decides online, that is, without delay, whether this signal is steady or not. This requires no complicated parameter assignment. In-depth process knowledge or detailed knowledge of the signal behavior is not necessary for the configuration.

### Configuration

The SteadyState block is dragged from the block catalog and dropped onto the specially prepared CFC. The selection of the insert position and the OB in which the block is installed depends on the insert position of the driver block for the process value to be monitored.

### Startup characteristics

Use `Feature.Bit0` (Setting the startup characteristics (Page 148)) to define the startup characteristics of this block.

- **Bit = 0:** The most recently stored values are reset at startup.
- **Bit = 1:** The most recently used values are sent to the following output parameters at startup:
  - `PV_Filtered`
  - `StatAct`

**Note**

Additional information about `Feature.Bit0` can be found in the section Setting the startup characteristics (Page 148).

### Status word assignment for the `Status1` parameter

For a description of the individual parameters, refer to the section SteadyState I/Os (Page 620).

| Status bit | Parameter |
|---|---|
| 0 – 2 | `Not used` |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `Not used` |
| 6 | `OnAct.Value` |
| 7 – 27 | `Not used` |
| 28 | `Internal error` |
| 29 - 30 | `Not used` |
| 30 | `StatAct.Value` |
| 31 | `NOT StatAct.Value` |

## 4.4.2 SteadyState modes

This block can be set to the following modes:

- On
- Out of service

### "On"

General information about the "On" operating mode can be found in the section "On (Page 77)" mode.

### "Out of service"

General information about the "Out of service" operating mode can be found in the section "Out of service (Page 77)" mode.

## 4.4.3 SteadyState operating principle

### Calling OBs

The cyclic interrupt OB into which you install the block (for example, OB32). Additionally in OB100 (see Setting the startup characteristics (Page 148)).

**Note**

The block must be installed in the same cyclic interrupt OB as the associated processing or monitoring block (e.g. MonAnL).

## Principle of Operation

The SteadyState calculation is implemented in four steps:

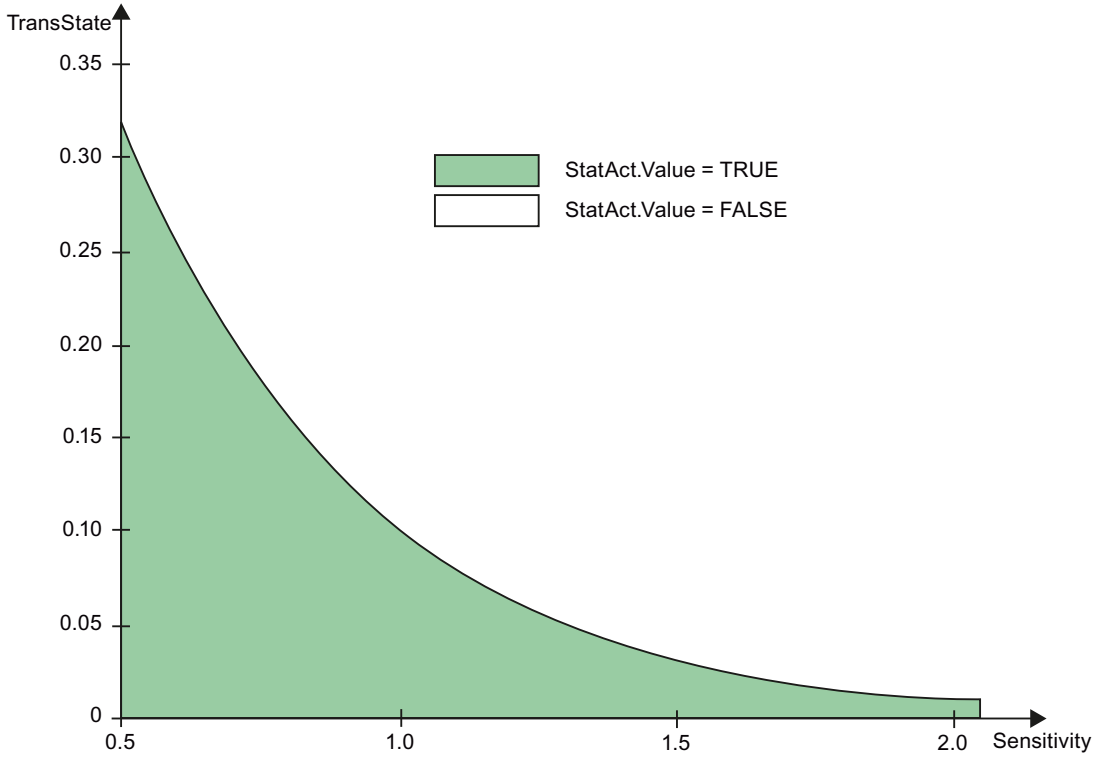
1. The input signal is filtered by a Kalman filter with the `TimeConstant` and displayed at the `PV_Filtered` output variable.
2. The signal is stored in a sliding time window with the length `TimeWindow`. Internally, the buffer length is limited to 1024 values. A signal is only classified as steady if it is steady throughout the entire time window.
3. The filtered signal in the sliding time window is analyzed for variances.
4. The result is provided in two forms: As a binary signal `StatAct` (`true/false`) and as an analog value `TransState`, which represents a scale for the unsteadiness (0 to 1, 0: steady, 1: completely unsteady).

The algorithm for detecting steadiness is based on a modified "Walsh transformation", in which the values in the sliding time window are tested for the presence of certain discrete frequencies.

The sensitivity of the steadiness detection can be adjusted using the `Sensitivity` tuning parameter. The default value is 1.

A steady state is detected when the following condition is satisfied:

$$TransState \leq 10^{-Sensitivity}$$




This makes it clear that with high sensitivity, the process value may move very little in order to be classified as steady.

## 4.4.4 SteadyState functions

### Detection of signal steadiness

The information as to whether a signal in the process control system is currently steady is useful for many applications.

**Applications from the field of process monitoring are:**

- The startup of steady processes is often characterized by fluctuating process variables. When these variables become steady, the process is in a steady state. Monitoring the steadiness of the relevant process variables therefore immediately indicates the end of the startup process. Example: End of the startup process of a pump when a steady flow is achieved.
- In steady processes, changing the product, a raw material or an input material results in fluctuating process variables. If the steadiness of the relevant variables is monitored, re-establishment of the steady state immediately indicates the start of the new process state. Example: Detection of a constant combustion temperature after fuel switching by monitoring the temperature of the combustion chamber or the exhaust.
- After disruption of the regular operation of a steady process, the process returns to its normal state. By monitoring the steadiness of the corresponding process variables, the cessation of the disruption is immediately detected. Example: Fluctuations in the inlet of a distillation column disrupt the head temperature. A return to a high-quality head product is indicated by the re-establishment of the steady state.

**Applications in the field of closed-loop and open-loop control are:**

- Time slices of the signal patterns containing a transition from one steady state to another steady state are useful for adjusting controller parameters. The detection of the steadiness of the variable to be controlled thereby serves to select the appropriate time slices. Example: Controller setting with SIMATIC PCS 7 PID Tuner or ModPreCon engineering tool.
- In batch processes, transition to the next basic operation in the recipe often depends on achieving certain process conditions. This situation can be indicated by the detection of the corresponding signal's steadiness . Example: A step sequence in the Sequential Function Chart (SFC) does not activate the agitator motor of a reactor for a fixed and possibly excessive period of time; instead, it stirs until the temperature in the reactor has a steady value.
- In general, switching the same process variables designed for different targets between different controllers may only be carried out in steady-state mode. This mode can be detected by monitoring the steadiness of the control variable. Only then can the switching of the controller be enabled. Example: Bumpless switching from a PI controller to a PID controller is only possible in a steady state in which the derivative component is zero.

### Detection of signal unsteadiness

The information as to whether a signal in the process control system is currently leaving steadiness has many applications.

**Applications from the field of the process monitoring are:**

- When a normally constant measured signal suddenly fluctuates, this points to a disturbance in the process or in the field device. The monitoring of the corresponding signal immediately indicates the disruption. Leaving steady state can even be detected before the signal reaches its warning or alarm limits. Example: The head temperature in the distillation column starts to fluctuate without violating the configured limits.
- Control loops designed to keep certain process variables constant can be monitored with the function block in a simple way. The detection of unsteadiness then means that the controller can no longer properly fulfill its task. Example: Monitoring of a PID controller for adherence to the steady state.
- Continuous units can be monitored for proper steady-state operation. As soon as the relevant process variables of a unit begin to fluctuate, drift or fail, the operator is notified or countermeasures are taken automatically. Example: If the head temperature of a distillation column is unsteady, the operator is immediately alerted to intervene accordingly.

## Determination of the plant status

The plant status is a criterion for a series of actions or the suppression of actions. The SteadyState function block distinguishes between "safe steady" and "not steady" states of plants or units. Applications for the binary signal in the control system are:

- Depending on the plant status, messages and alarms can be activated, suppressed or hidden. In this way, alarm surges of transient irrelevant messages can be prevented. Example: Alarm messages from the lower-level controllers are suppressed in unsteady startup or product changeover phases.
- Limits of function blocks can be changed depending on the plant status. Strict limits are often more suitable in steady-state operation than in unsteady operation. This approach considerably reduces alarm surges if a malfunction occurs. Example: A temperature value is monitored for a few degrees of deviation in steady-state normal operation. For startup phases, however, it is monitored only in accordance with the field device specification.
- Controllers or controller parameters can be selected depending on the plant status. Highly responsive controllers, which are generally not robust, are suited for steady-state operation within strict limits. In contrast, it is advisable to switch to a slower but more robust controller for startup phases or in cases of malfunction. Example: A temperature controller for startup processes is used alternately with a temperature controller for steady phases.
- Step sequences can use the current plant status, steady or not steady, as a condition for forwarding. Example: Simply reaching the setpoint temperature in a heated tank is not an adequate criterion for further action; the temperature must also be steady at this value.

## Forming the signal status for blocks

The signal status of the input signal (`PV.ST`) has no influence on the behavior of the function block.

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `PV.ST`

### Selecting a unit of measurement

This block provides the standard function "Selecting a unit of measure (Page 240)".

### Configurable reactions using the "Feature" parameter

An overview of all the reactions that are provided by the `Feature` parameter is available in section "Configurable response using the Feature I/O (Page 139)". The following modes of behavior are available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Setting the startup characteristics |
| 1 | Characteristics for the out of service mode |

### Operator control permissions

This block provides the standard function "Operator control permissions (Page 139)".

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 – 6 | Not used |
| 7 | 1 = Operator can reset the steadiness detection |
| 8 – 12 | Not used |
| 13 | 1 = Operator can change the filter time constant |
| 14 | 1 = Operator can change the time horizon |
| 15 | 1 = Operator can change the sensitivity |
| 16 | 1 = Operator can change the amplitude band |
| 17 – 31 | Not used |

### SIMATIC BATCH functionality

This block does not support "SIMATIC BATCH functionality".

### See also

Operator control permissions (Page 285)

## 4.4.5 SteadyState error handling

### Overview of error numbers

The `ErrorNum` I/O can be used to output various error numbers:

| Error ID | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 2 | `SampleTime` parameter is too small (< 0.001) |
| 30 | The value PV can no longer be displayed in the REAL number field. |

## 4.4.6 SteadyState I/Os

### I/Os of SteadyState

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `AmplBand` | Positive number that indicates the amplitude band, i.e. the range in which the observed signal varies. It is useful to specify the amplitude band if this range of variation is small compared to the absolute values of the observed signal. Otherwise, the function block monitors the absolute amplitude whereby, for example, temperature fluctuations between 418 and 422 °C are very small and therefore always appear to be steady. If, however, such fluctuations are relevant, an amplitude band of 4 would be configured in this example. | REAL | 0.0 |
| `BlkTypeInfo*` | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| `EN` | 1 = Called block is executed | BOOL | 1 |
| `Feature` | I/O for other functions (see SteadyState functions (Page 617)) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| `OnOp*` | 1 = "On" mode via operator | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OosLi | 1 = "Out of service", via interconnection or SFC (0–1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator input | BOOL | 0 |
| OS_Perm | I/O for operator control permissions (see SteadyState functions (Page 617)) | STRUCT<br>• Bit0: BOOL<br>• Bit10: BOOL<br>• Bit31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| PV | Process value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_OpScale | Limit for PV trend display of the faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| PV_Unit | Unit of measure for process value | INT | 1001 |
| RstOp* | 1 = Reset steady state monitoring.<br>This function block immediately switches to unsteady at the output. | BOOL | 0 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| Sensitivity | Positive number that indicates the sensitivity of the monitoring to fluctuations in the input signal. It must be between 0.1 and 10, whereby 1 is the normal value. However, even the values ½ or 2 represent a very significant change in the sensitivity; values below or above are suitable for extreme applications at best.<br>"Values greater than 1 mean that even minor changes in the signal will be classified as "unsteady"". | REAL | 1.0 |
| TimeConstant | Positive number in seconds which indicates the time constant of the low-pass filter. It is limited internally to a maximum of 10,000 sample steps based on the SampleTime parameter. A low-pass filter is applied to the input signal with the specified time constant and sent to the PV_Filtered output. | REAL | 5.0 |
| TimeWindow | Positive number in seconds that specifies how long steadiness must be observed before the block sets the corresponding StatAct output. High values thus lead to long wait times before steadiness is detected.<br>It is limited internally to a maximum of 1,024 sample steps depending on the SampleTime parameter. If more than 1,024 sampling steps are needed to monitor the signal, a clock reduction ratio is required. | REAL | 50.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of the existing error number. For the error numbers that can be output by this block, see section SteadyState error handling (Page 620) | INT | -1 |
| MeanValue | Analog signal that outputs the filtered input signal averaged over the time horizon. This is the running average of the filtered input in the time window that has been selected for the TimeWindow parameter. | REAL | 0.0 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| PV_Filtered | Analog signal that outputs the filtered input signal. The filter is a low-pass filter with a time constant that has been selected for the TimeConstant parameter. | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_OpScaleOut | PV_OpScale as output | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| PV_UnitOut | PV_Unit as output | INT | 0 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| StatAct | A logic signal that outputs 1 if it detects steadiness of the input, otherwise 0. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Status1 | Status word 1 | DWORD | 16#00000000 |
| TransState | Analog signal between 0 and 1, which is a measure of the unsteadiness of the signal. The closer the signal is to 0, the steadier the signal is assessed to be. | REAL | 1.0 |
| Trend | Analog signal that indicates the time derivative of the filtered input signal in the steady state, averaged over the time window. The signal is constantly 0 for non-steady input signals.<br>A trend in this respect is a slow creep, which is not yet classified as unsteady. | REAL | 0.0 |

## 4.4.7 SteadyState block diagram

A block diagram is not provided for this block.

## 4.4.8 Operator control and monitoring

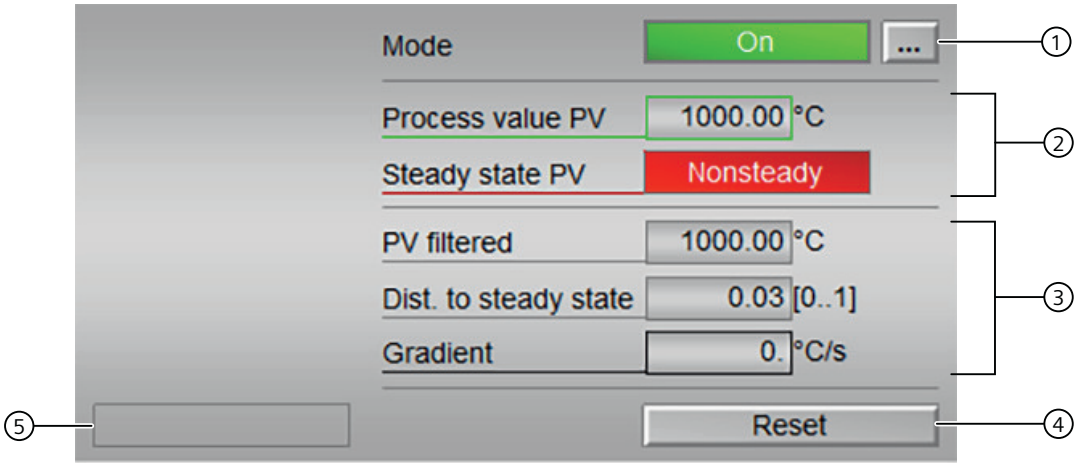
### 4.4.8.1 Views of SteadyState

The SteadyState block provides the following views:

- Standard view of VlvMon (Page 623)
- Parameter view of SteadyState (Page 624)
- Trend view of SteadyState (Page 625)
- Memo view (Page 338)
- Block icon for SteadyState (Page 625)

For general information on the faceplate and block icon, refer to the "Structure of the faceplate (Page 279)" and "Block icon structure (Page 260)" section.

### 4.4.8.2 SteadyState standard view




**(1) Display and switchover of the operating mode**

This area provides information on the operating mode currently in effect. The following operating modes can be shown here:

- On
- Out of service

**(2) Process value**

This area shows the current process value `PV` and its steady status. The status display can have the following states:

- "Steady" (green)
- "Unsteady" (red)
- No text (grey)

**(3) Further characteristic values**

In this area, the filtered process value (`PV_Filtered`), the scale of unsteadiness (`StatState`) and the gradient (`Trend`) are displayed. The values cannot be operated.

**(4) Resetting the steadiness detection**

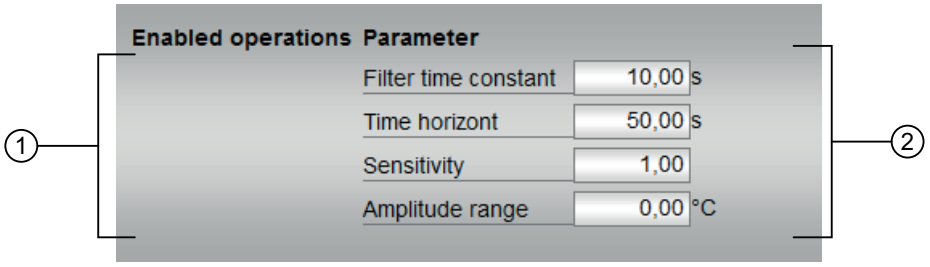
Pressing the Reset button resets the entire steadiness detection.

**(5) Display area for block states**

This area provides additional information on the operating state of the block:

- "Internal error"
- "Invalid Signal"

### 4.4.8.3 SteadyState parameter view




**(1) Enabled operations**
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark:** The OS operator can control this parameter
- **Gray check mark:** Access to this parameter is temporarily disabled for the OS operator due to process conditions
- **Red cross:** The OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm`).

**(2) Parameters**
The following parameters can be entered in this area:

- "Filter time constant": Time constant of the low-pass filter (`TimeConstant`)
- "Time horizon": Length of the time window for monitoring the signal (`TimeWindow`)
- "Sensitivity": Sensitivity of the monitoring to variations of the input signal (`Sensitivity`)
- "Amplitude band": Amplitude band for the monitoring (`AmplBand`)

### 4.4.8.4 SteadyState trend view




The trend view displays the process value `PV` (dark green), the filtered process value `PV_Filtered` (blue) as well as the status of the signal `StatAct` (green-red).

### 4.4.8.5 Block icon for SteadyState

A block icon with the following displays is available to you:

- Block type or process tag
- Operating modes
- State of the monitored signal

The block icon from template @TemplateAPLV9.PDL:

| Symbols | Selection of the block icon in CFC | Special features |
|---|---|---|
| | 1 | Block icon in full display in steady state |
| | 1 | Block icon in full display in non-steady state |
| | 1 | Block icon in the "Out of service" operating mode |
| Steady | 2 | Block icon in full display in steady state |
| Nonsteady | 2 | Block icon in full display in non-steady state |
| | 2 | Block icon in the "Out of service" operating mode |

You can find additional information about the block icon and the operator control options in the block icon in the following sections:

- "Configuring the block icons (Page 267)"
- "Block icon structure (Page 260)"
- "Operation via the block icon (Page 269)".

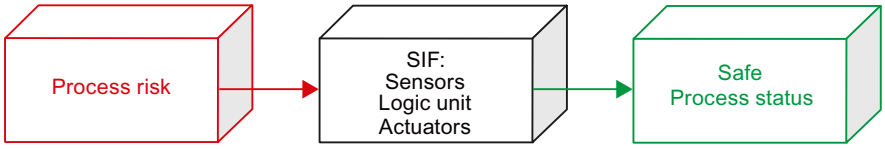
# 4.5 Partial Stroke Test (PST)

## 4.5.1 Introduction

### 4.5.1.1 What is the Partial Stroke Test?

A Safety Instrumented Function (SIF) can be described as a function with which a specific Safety Integrity Level (SIL) is implemented with regard to a specific process risk. A Safety Instrumented Function normally consists of sensors, a logic unit and actuators. Only when the components of the Safety Instrumented Function operate together is the risk detected, then the process is set to a safe status. To ensure that the Safety Instrumented Function (SIF), where required, in fact functions as intended, your technical systems must be tested regularly.

A Safety Instrumented Function (SIF):




The Safety Integrity Level (SIL) achieved is derived from the average Probability of Failure on Demand (PFD) for an SIF. Failure on demand is when a device does not perform the intended function when requested. This probability of failure is defined as PFD. The PFD value is calculated on the basis of the rate of failure (λ) and the proof test interval (TI) of the device. If only the service life of one device is considered, a constant λ value can be assumed. As λ therefore has a constant valve for a given valve, the relationship between PFD and TI is directly proportional. The following applies: The greater the TI value, the greater the PFD value. This means that more frequent tests keep down the PFD value and ensure that a Safety Instrumented Function achieve the Safety Integrity Level that has been defined for it.

The actuators of a Safety Instrumented Function generally contribute most to the overall PFD value. During normal plant operation, safety shutoff valves normally stay in just one position: fully open or fully closed. These safety valves only have to perform a movement in an emergency. The fact that they do not move for long periods of time can cause the valves to block under the tough plant conditions. Regular tests are therefore required to ensure that the valve is functioning correctly and is not blocked in an emergency situation.

To test all the failure modes of a safety shutoff valve completely, it is necessary to ensure that the valve can be set to a fail-safe state. This type of test is referred to as a Full Stroke Test (FST). A Full Stroke Test normally tests the valve drive, the valve seat and, where one exists, the solenoid valve. If e.g. sealing and a short response time are required for a safety shutoff valve, these properties should also be tested in a Full Stroke Test.

Many safety shutoff valves can be tested online using a physical valve bypass. For other valves, however, the Full Stroke Test requires the process to be interrupted and causes an unwanted valve switching procedure. In these cases, the Full Stroke Test can only be performed during plant shutdown or planned maintenance when the valve can be completely opened and closed. Otherwise, costly procedures are required, e.g. using and independently testing multiple parallel valves.

A Partial Stroke Test makes it possible to guarantee that a valve is not blocked, even when it stays in normal position, without negatively affecting the process or causing incorrect tripping. In this test, the valve normally completes a stroke of 10-20% of the full stroke. The actual stroke value of the valve depends on factors such as valve types and size, the manufacturer's recommendations and the operating conditions. After the valve has been tested in test position, it is reset to normal state. A Partial Stroke Test cannot test all the failure modes of a safety valve, but only the failure modes relating to the valve drive.

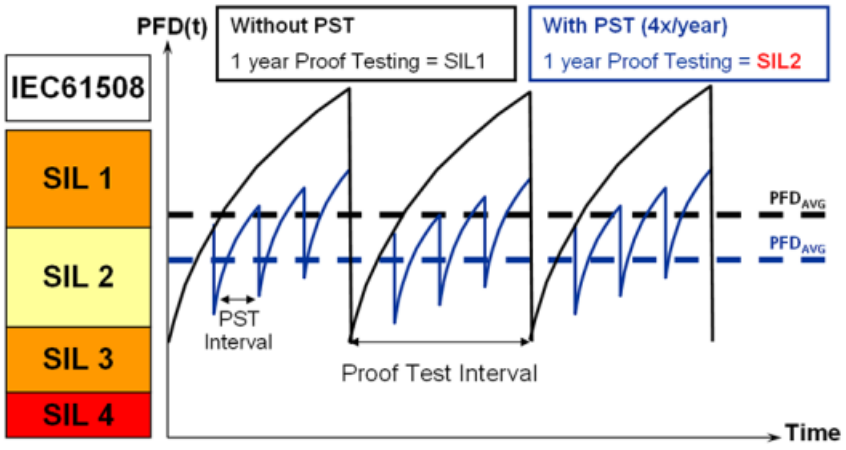
The most common error patterns in valve drive systems are listed in the following table.

| Fault description | Possible cause of error | Fault detectable? | | Fault detection |
|---|---|---|---|---|
| | | Partial Stroke Test | Full stroke/ full rotation test | |
| Solenoid valve not switching | Trip for the solenoid valve defective | Detectable | Detectable | Detection via feedback value |
| Solenoid valve not switching | Solenoid valve defective | Detectable | Detectable | Detection via feedback value |
| Valve reacts too slowly | Air line to the valve is pinched | Detectable | Detectable | Detection by monitoring the time until feedback value is given |
| Valve reacts too slowly | Valve is sluggish | Detectable | Detectable | Detection by monitoring the time until feedback value is given |
| Valve does not close or does not close completely | Valve seat pitted<br>Disk eroded | **Not** detectable | Detectable | Detection not possible via Partial Stroke Test |
| Valve does not close or does not close completely | Valve set has deposits | **Not** detectable | Detectable | Detection not possible via Partial Stroke Test |
| Valve does not close | Valve stem blocked | Detectable | Detectable | Detection via feedback value |
| Source: NAMUR Recommendation NE106, first published: 01/03/2006, Test Intervals for Safety Integrated Systems, Table 3: Most common error patterns in valve drive systems | | | | |

The diagnostic Coverage of a Proof Test (CPT) is a percentage measure for the effectiveness of a Partial Stroke Test. If the CPT value for a safety shutoff valve is 70%, for example, a Partial Stroke Test can detect 70% of all possible failure modes of the valve. A Partial Stroke Test, however, cannot test all failure modes of a safety valve, but only demonstrate that a valve is not blocked and thereby reduce the total PFD value. This contributes to maintaining the Safety Integrity Level (SIL).

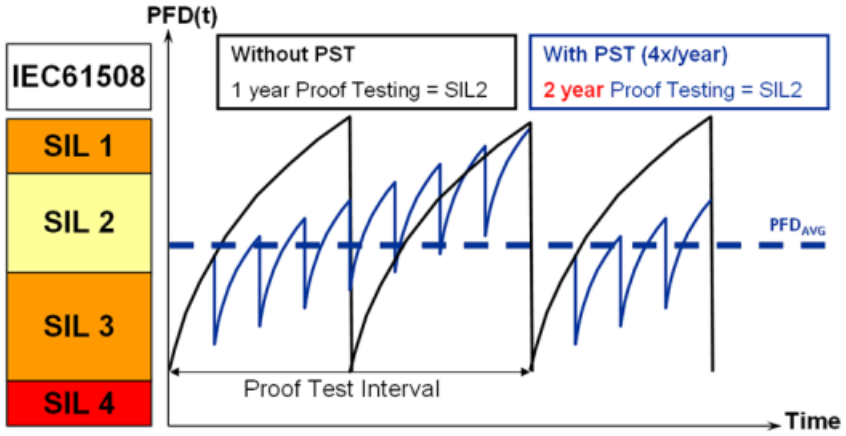
The following table shows how the SIL for a Safety Instrumented Function is increased using a Partial Stroke Test by lowering the PFD value for a safety shutoff valve. The following figure shows how the safety valve only reaches SIL 1 with an annual test interval for a Full Stroke Test (Proof Test) (based on the average PFD value and the IEC 61511 guidelines). By contrast, the safety valve reaches SIL 2 with an additional Partial Stroke Test at intervals of 3 months.

Using the Partial Stroke Test to achieve a higher SIL:




The following figure shows how the required interval for the Full Stroke Test (Proof Test) can be extended by using Partial Stroke Tests, without changing the SIL. The following figure shows how the safety valve achieved SIL 2 with a Full Stroke Test interval of one year. If a Partial Stroke Test is performed at intervals of 3 months, the Full Stroke Test interval can be extended to 2 years and SIL 2 can still be achieved.

Using the Partial Stroke Test to increase the Full Stroke Test interval:




**Note**

As a Partial Stroke Test does not provide a full function check of a valve, a Proof Test must still be performed at a predefined test interval.

### 4.5.1.2 The Partial Stroke Test application

The Partial Stroke Test application contributes to the safe and reliable operation of safety shutoff valves. The application uses the S7-400 F/FH as part of the SIMATIC safety system in conjunction with an integrated valve position controller to test a pneumatic shutoff valve. Under the Partial Stroke Test solution, the position controller is used to close the valve partially to demonstrate that the valve has moved to the required position and then to reset the valve to its open position. **The safety function itself is guaranteed by a fail-safe discrete output.**

**By defining a setpoint** for the position controller **and as a result of fail-safe discrete shutoff, two separate signals are available to activate the valve's safety position**, delivering a "fail-safe" 1oo2 evaluation. According to the published report of an independent consultancy as a third-party provider, the hardware layout for the Partial Stroke Test is SIL3-capable. 2

The method used in the Partial Stroke Test application for the Partial Stroke Test is referred to as position control. It can be used for rotary slide and screw-down valves and requires the installation of an intelligent valve position controller together with a shutoff valve.

---

**Note**

This application can also be used for other drives (e.g. motor) if a feedback is available.

This application can also be used for Partial Stroke Tests even without position controller if:

1. the discrete shutoff is also actuated by the test logic
2. the binary feedback is used to simulate an analog feedback.

---

The Partial Stroke Test application contributes to the safe and reliable operation of safety shutoff valves. The application uses the S7-400 F/FH as part of the SIMATIC safety system in conjunction with an integrated valve position controller to test a pneumatic shutoff valve. Under the Partial Stroke Test solution, the position controller is used to close the valve partially to demonstrate that the valve has moved to the required position and then to reset the valve to its open position. The test is protected by a fail-safe discrete output. The output provides an independent ability to return the valve to its fail-safe position.

The analog output for the position controller in conjunction with the discrete output supplies two separate signals for safety shutoff to the drive, which together provide "fail-safe" 1oo2 evaluation. According to the published report of an independent consultancy as a third-party provider, the hardware layout for the Partial Stroke Test is SIL3-capable.

---

**Note**

The hardware report has been created by Exida and is available on request from Siemens. The applicable document details are as follows:

- Safety Integrity Level Verification
- Project: Final Element Configurations, Report No.: SIE 04/08-20 R001
- Version V1, Revision 1.0, October 2004
- William Goble – Rachel Amkreutz

---

The method used in the Partial Stroke Test application for the Partial Stroke Test is referred to as position control. It can be used for rotary slide and screw-down valves and requires the installation of an intelligent valve position controller together with a shutoff valve.

The PCS 7 Operator Station controls the shutoff valve during a Partial Stroke Test and records and saves the test results for documentation purposes and for prospective maintenance.

All types of valve position controllers are suitable for use with the Partial Stroke Test application. The safety component of the application defines the setpoint for the valve position and compares it to the feedback of the valve position at the time of the test.

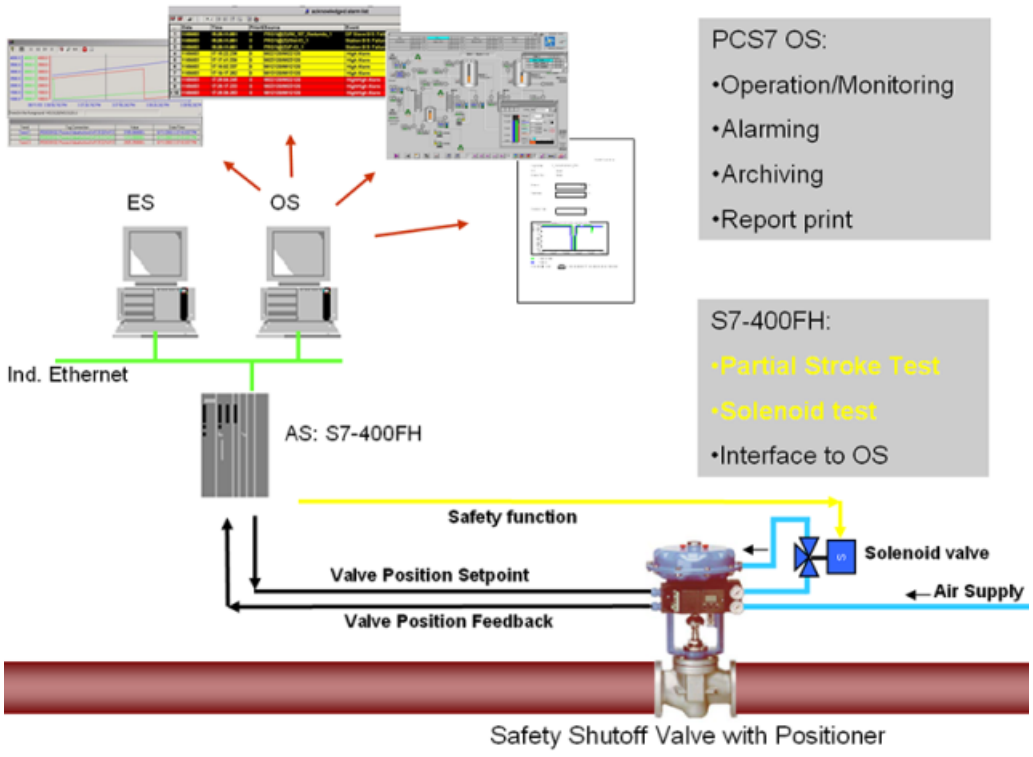
The Partial Stroke Test application includes three components (see also the following figure):

- **Partial Stroke Test engineering template:** Consists of a STEP 7 engineering template (i.e. pre-configured logic) to set up, run and monitor a Partial Stroke Test and, optionally, a solenoid valve test. This engineering template contains a CFC "chart-within-a-chart", which contains CFC charts to run the Partial Stroke Test at configured intervals. This program logic also includes PFD calculations to forecast the time period until the next required Full Stroke Test. The template has an interface to the PCS 7 Operator Station.
- **Partial Stroke Test user interface:** Consists of a block icon and a faceplate for the PCS 7 OS, visualizes the operation and status of the Partial Stroke Test and offers integrated alarm messages and event logging.
- **Partial Stroke Test test document:** Provides a pre-configured report layout for the PCS 7 OS which automatically documents the Partial Stroke Test as a printout.

The Partial Stroke Test application has the following benefits:

- Test safety shutoff valves with regard to various different failure modes
- Feedback for data that can be used to forecast future possible valve failures
- More flexible possibilities for testing and longer proof test interval
- Lowers the PFD value of a safety shutoff valve
- Alarm message when a Partial Stroke Test has failed or been inhibited and when a Full Stroke Test is required to achieve the specified SIL.
- Printout of test results
- Choice between manual and automatic tests

Overview of the Partial Stroke Test with S7-400FH and PCS 7:



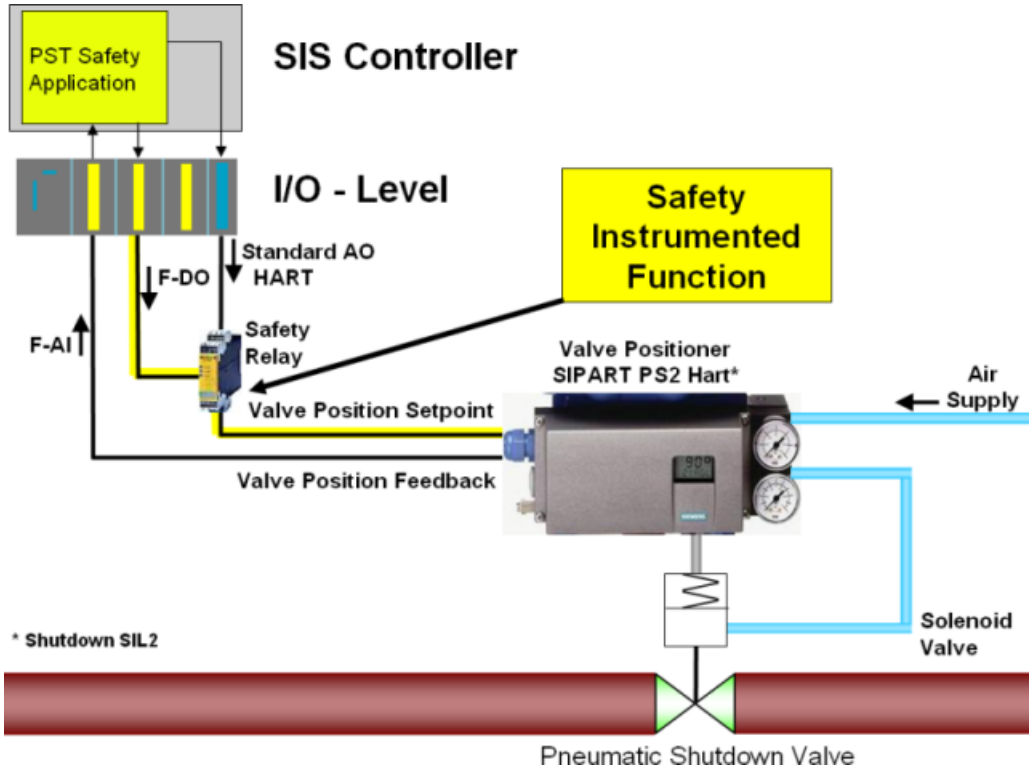

### 4.5.1.3 Hardware overview

An S7-400 F/FH-CPU in conjunction with an intelligent valve position controller that supplies analog feedback data on the valve position is used for the Partial Stroke Test application. Valves that are actuated by stored energy are intended for use with the application. The valves are valves that are normally open and designed to close in the event of safety shutoff.

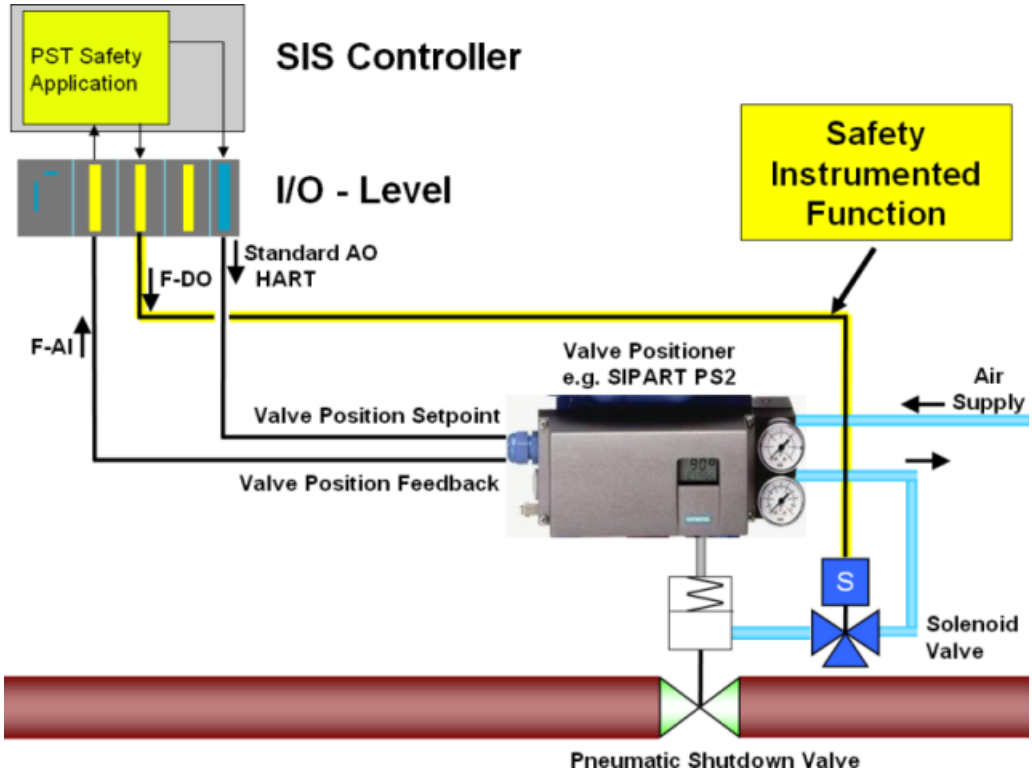
The following figures show two proposed arrangements of the physical system using a permanently wired valve position controller.

- Standard analog output (or standard analog output with HART) for a signal that transfers the desired valve position setpoint to the valve position controller.
- Fail-safe analog input for a signal that monitors the actual valve position
- Fail-safe discrete output for a signal that closes the valve in the event of the conditions for safety shutoff. The two possible configuration are shown in the following figures:
  - Safety relay: Interrupts the power supply to the valve position controller so that the controller interrupts the supply of compressed air to the valve

– Solenoid valve: Exhausts the air from the valve so that the spring closes the valve
Partial Stroke Test with SIPART PS2 HART for shutoff:



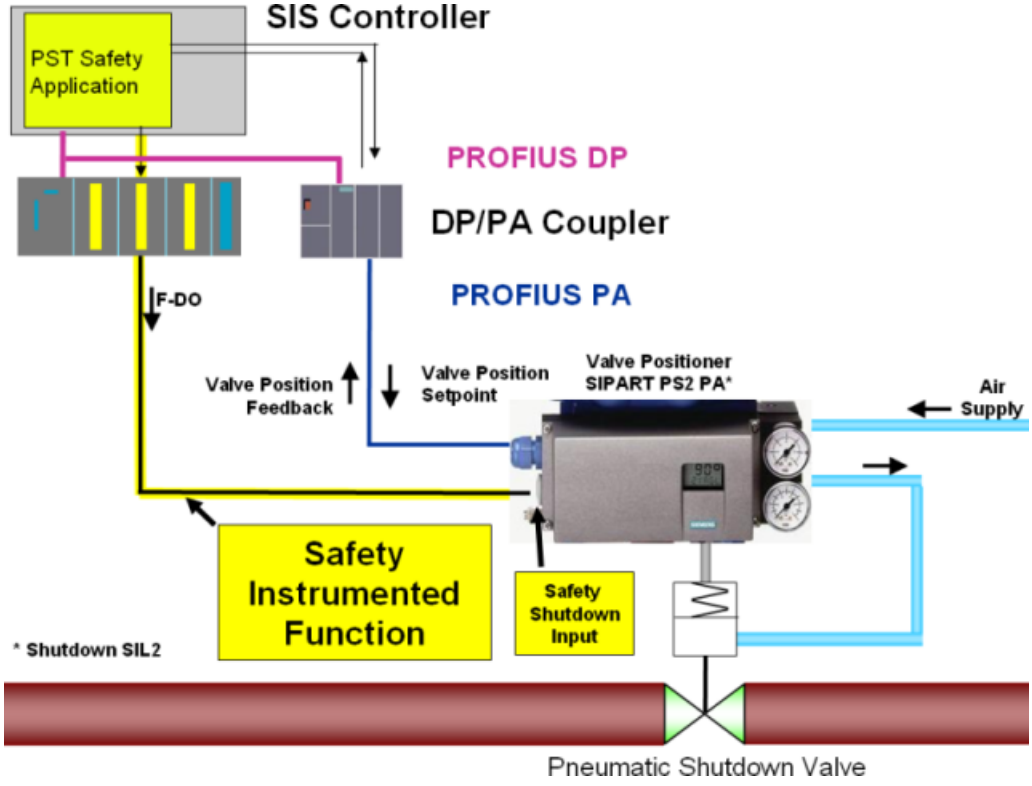

Partial Stroke Test with HART position controller for diagnostics only:


PST Safety Application
SIS Controller
I/O - Level
Safety Instrumented Function
Standard AO HART
F-DO
F-AI
Valve Positioner e.g. SIPART PS2
Valve Position Setpoint
Valve Position Feedback
Air Supply
S
Solenoid Valve
Pneumatic Shutdown Valve

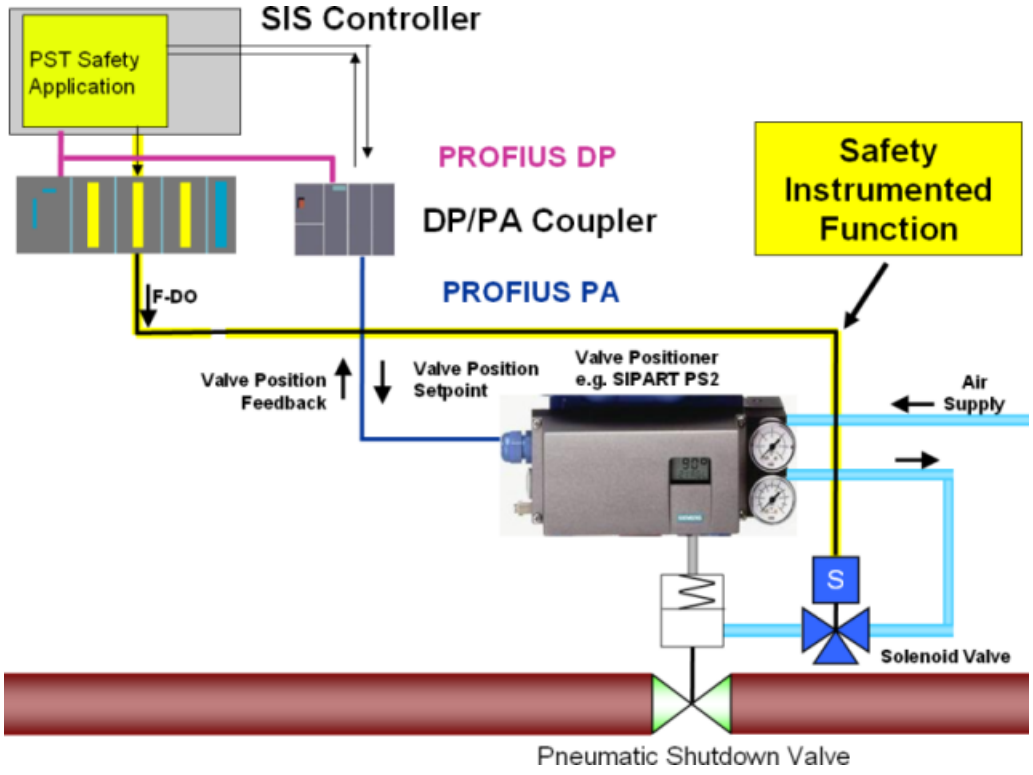
The following two figures show two proposed layouts of the physical system using position controllers and a fieldbus.

- The fieldbus transfers the desired valve position setpoint to the valve position controller and reads out the actual valve position
- Fail-safe discrete output for a signal that closes the valve in the event of the conditions for safety shutoff. The two possible configuration are shown in the following figures:
  - Input for safety shutoff: Supplies a switch-off signal to the valve position controller so that the controller interrupts the supply of compressed air to the valve

– Solenoid valve: Exhausts the air from the valve so that the spring closes the valve
Partial Stroke Test with fieldbus position controller for shutoff:




Partial Stroke Test with fieldbus position controller for diagnostics only:



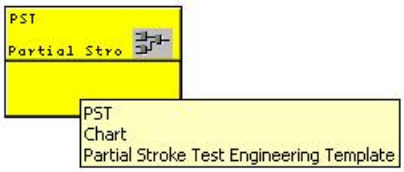

Installing a position controller in a Safety Instrumented Function does not automatically render a solenoid valve unnecessary. Many applications with safety shutoff valves require a solenoid valve because ventilation by the position controller is too low to close the valve quickly. If a solenoid valve is used, it should be installed between the valve position controller and the valve drive.

If both a position controller and a solenoid valve are used, the Partial Stroke Test does not test the solenoid valve. The Partial Stroke Test application, however, includes program logic as an option to test the solenoid valve. It is up to the operator to decide whether a solenoid valve test is necessary to test the solenoid valve function at the time of the Partial Stroke Test. The failure modes of the solenoid valve frequently do not have a significant impact on the overall PFD value of the valve body. If the solenoid valve is not a significant factor, it can be test as part of the normal proof test as part of the Full Stroke Test.

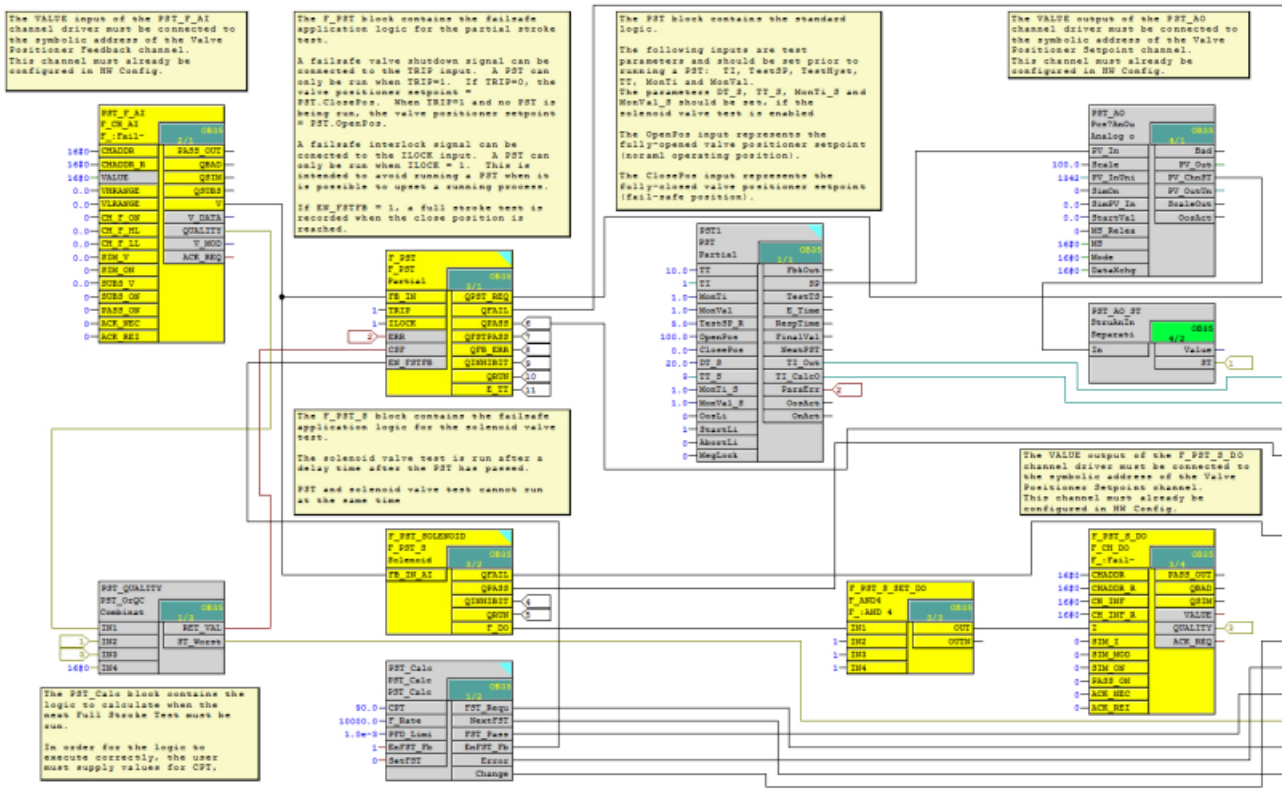
### 4.5.1.4 Overview of engineering template for Partial Stroke Test

The Partial Stroke Test engineering template component consists of pre-configured STEP 7 CFC logic. The logic is encapsulated in an engineering template (a CFC "chart-within-a-chart") that can be inserted into a CFC chart. The operator instructions in the template should help the operator understand and carry out Partial Stroke Test configuration.

Partial Stroke Test engineering template (encapsulated logic):




Partial Stroke Test engineering template (detailed logic):



The following function blocks and functions have been created to execute the Partial Stroke Test logic:

- PST: Contains standard logic of the Partial Stroke Test; compiled SCL code
- F_PST: Contains the fail-safe logic of the Partial Stroke Test; compiled CFC charts exclusively with TÜV-certified fail-safe system function blocks
- F_PST_S: Contains the optional fail-safe logic of the solenoid valve test; compiled CFC charts exclusively with TÜV-certified fail-safe system function blocks
- PST_Calc: Contains PFD calculations to determine the time of the next Full Stroke Test; compiled SCL code
- PST_OrQC: Contains logic that evaluates the signal status of a maximum of four channel driver function blocks to determine whether there is a poor-quality signal; compiled SCL code

### 4.5.1.5 Overview of Partial Stroke Test user interface

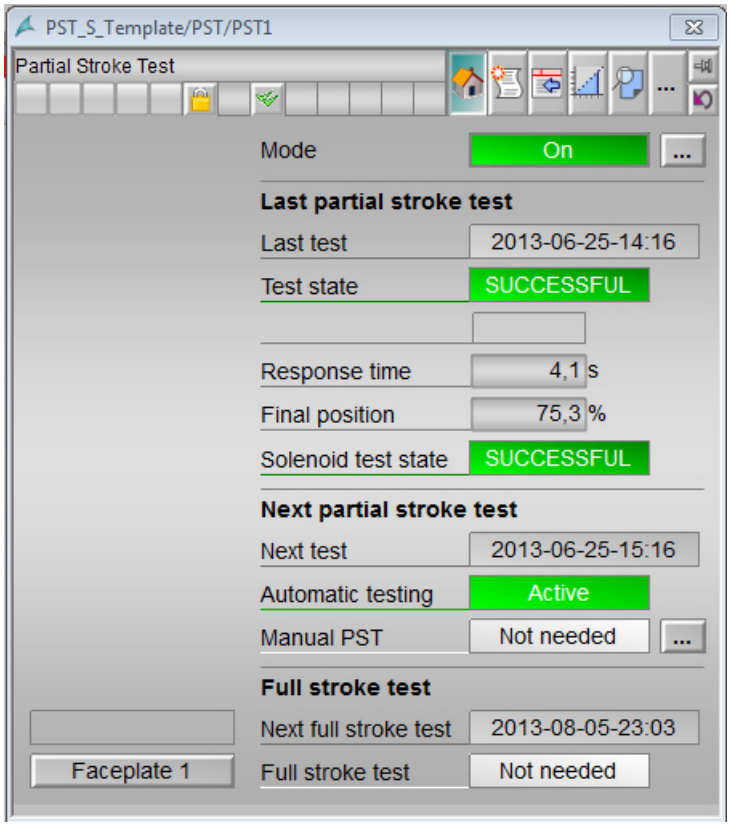
The user interface component of the Partial Stroke Test consists of a block icon and a faceplate for the PCS 7 OS. The block icon and the faceplate can be used to visualize the Partial Stroke Test process from a PCS 7 OS console and to monitor the status and the messages for an instance of the Partial Stroke Test as well. The operator can make detailed changes to the configuration parameters and manually start tests from the faceplate with the corresponding security permissions (see AUTOHOTSPOT).

To facilitate configuration, the block icon and the faceplate are created and configured automatically at a PCS 7 OS compile when the user interface and engineering template components are used in combination.

Block icon for Partial Stroke Test:




Faceplate for Partial Stroke Test (standard view):



## 4.5.2 Usage guidelines

### 4.5.2.1 Basic rules

#### (1) Purpose of this manual

This manual provides a complete overview of the Partial Stroke Test application. The manual is intended for qualified personnel familiar with creating programs for Partial Stroke Tests and for employees who are required to configure, operate and maintain automation systems.

#### (2) Basic knowledge required

Working with this manual requires general knowledge of automation engineering. The user should be trained in the following SIMATIC technologies used in the application: STEP7, CFC, S7 F/FH systems and PCS 7 OS. The user should also be familiar with computers or devices with similar functionality (e.g. programming devices) under Windows OS.

#### (3) What is covered by this manual?

This manual applies to the Partial Stroke Test application as a component of the Condition Monitoring Library.

#### (4) Qualified Personnel

This device must only be assembled and operated by qualified personnel. Qualified personnel within the meaning of this manual are persons who are authorized in accordance with safety engineering practices and standards to operate, ground and label electric circuits, equipment and systems.

### 4.5.2.2 Safety instructions

#### (1) Certificates and approvals

As per the corresponding TÜV report, the Partial Stroke Test application operates problem-free with the Safety Instrumented Function in the proposed physical system layouts. The hardware report has been created by Exida and is available on request from Siemens.

As the setpoint for the valve position is defined by the safety instrument components of the application and compared at the time of the test to the feedback for the valve position, the test sequence is therefore safer than with a component integrated into the position controller.

The user can configure the Partial Stroke Test application to operate with safety valves by using the instructions and information in this manual. The user is responsible for checking the Partial Stroke Test logic setup with regard to the safety valves and for calculating the necessary test intervals with which a specific SIL and/or an extended proof test interval should be achieved.

### (2) Observance of the safety instructions for SIMATIC S7 F/FH Systems

The Partial Stroke Test engineering template is an engineering template based on SIMATIC S7 F/FH Systems. The user must therefore real, understand and observe all the safety instructions in the "S7 F/FH Systems - Configuration and Programming. Programming and Operating Manual".

### (3) Notes on using intelligent valve position controllers with up/down valves

If the valve is set to a fixed actuation pressure, e.g. 100 %, and is blocked e.g. due to encrustation, minor leakages may under some circumstances lead to a pressure drop in the drive. If the pressure drop is large enough to break away the encrustation, there is a risk that the critical stroke value of the valve will be exceeded.

This risk can be detected and/or avoided using the following procedures:

- The blocked valve can be detected by the Partial Stroke Test application. Sufficiently short test intervals are a precondition in this regard.
- Feeding the compressed air directly into the drive (setting in the intelligent valve position controller) prevents a pressure drop as a result of minor leakages.
- Use a pressure sensor to detect a critical (reduced) pressure.

## 4.5.3 Getting Started

### 4.5.3.1 Hardware requirements

The following hardware is required to use the Partial Stroke Test application software with PCS 7:

- A PC with CD-ROM drive
- The hardware components of the SIMATIC S7 F/FH Systems
  - S7 F/FH System CPUs (z. B. CPU417-4H) with fail-safe system license
  - Fail-safe signal modules (F-SM)

The Partial Stroke Test application is designed for valves that operate on the quiescent current principle (i.e. valves that are open in normal condition). To run the Partial Stroke Test application as intended, the following field instrumentation is also required:

- A valve position controller with analog feedback
- A safety relay or solenoid valve

**Note**

If using a relay instead of a solenoid valve, a safety relay is required under some circumstances, depending on the required SIL for the Safety Instrumented Function.

**Note**

As of the current version, the Partial Stroke Test application only supports valves that are open in normal condition (i.e. valves that are closed in fail-safe state).

#### 4.5.3.2 Software prerequisites

The following software is required to be able to use the full scope of the software for the Partial Stroke Test engineering template and the user interface (details are provided in sections below):

The general prerequisites for SIMATIC PCS 7 apply.

**(1) Partial Stroke Test engineering template**

- S7 F systems V6.2 Upd1 and higher
- S7 F library V1.3 SP2
- PCS 7 V9.0

**(2) Partial Stroke Test user interface**

PCS 7 OS V9.0 and higher

### 4.5.4 Configuration of the engineering template

#### 4.5.4.1 Overview of Partial Stroke Test logic

The Partial Stroke Test engineering template component consists of pre-configured CFC logic. To simplify configuration, this logic has been encapsulated in an engineering template (a CFC "chart-within-a-chart") that the user can drag and drop onto a CFC chart. The user then only has to connect the I/O channel to be used (for the feedback from the valve etc.) and specify the configuration parameters.

The engineering template for the Partial Stroke Test has the following basic performance features:

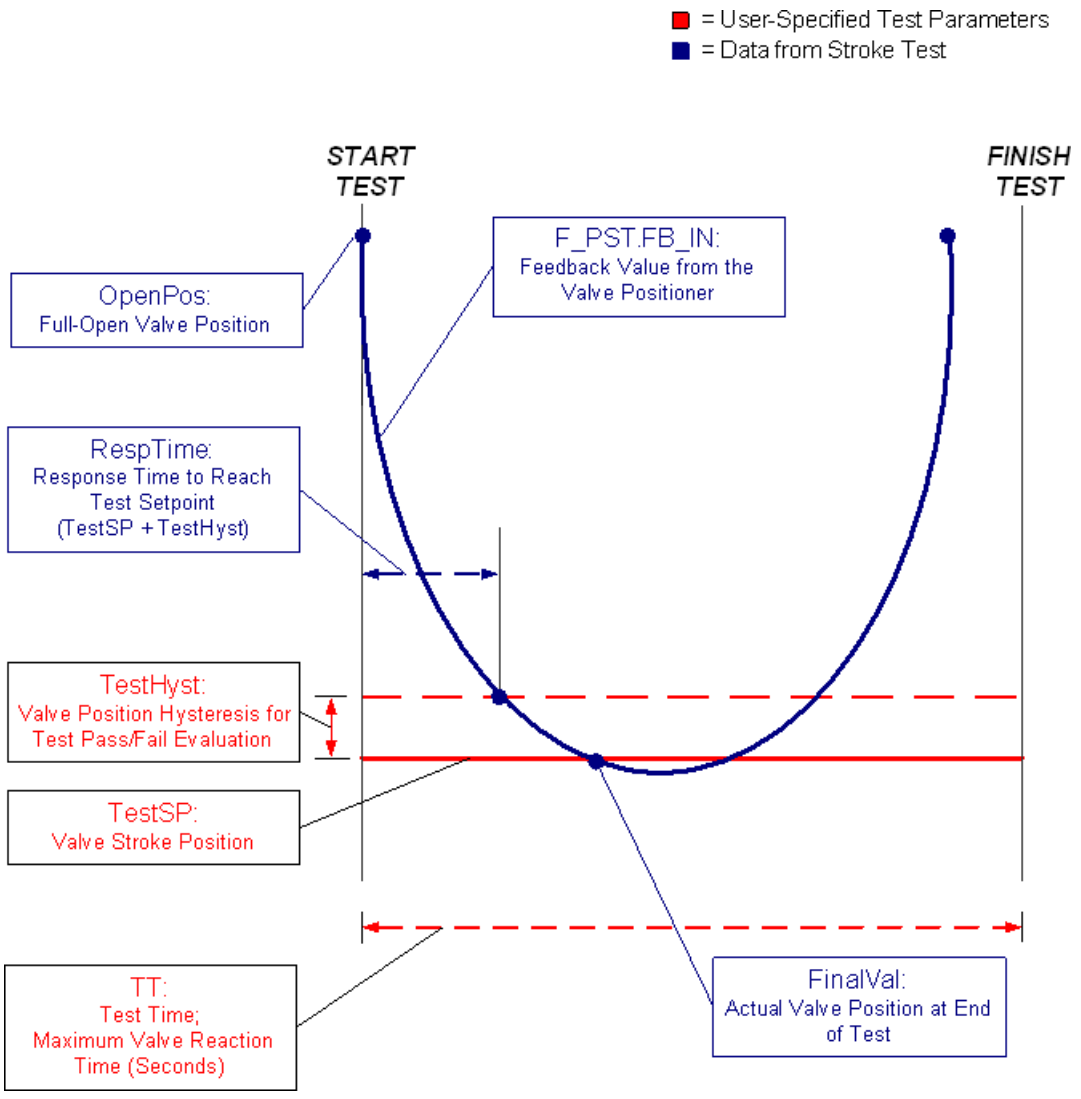
- Automatic running (at time intervals defined by the user) or manual running of Partial Stroke Tests. The system can issue a warning that a test should be run. Manual test can be started from the CFC logic itself or via the corresponding PCS 7 OS faceplate.
- Detection of whether a Partial Stroke Test, Full Stroke Test or a test outside the logic of the Partial Stroke Test application (e.g. locally at the valve position controller) has been run.
- The user can configure a valve setpoint for the test and a hysteresis used to determine the success/failure status of a Partial Stroke Test.
- The user can define the maximum time interval available to pass a Partial Stroke Test or solenoid valve test.
- The user can define a time delay between a passed Partial Stroke Test and a solenoid valve test.
- Various alarm messages that can be displayed in PCS 7 OS. Alternatively, messages can also be disabled via a disable option if no HMI is used for the Partial Stroke Test application logic

- The option for the user to monitor the valve performance. The following information is displayed for each Partial Stroke Test for this purpose:
  - Time/date of the start of test
  - Success/failure status of the test
  - Time taken to reach the desired valve position
  - Actual valve position at end of test
- Activation of the interlocking and logical enabling signals to inhibit stroke tests in cases where they are not wanted (if signal is forwarded or low signal quality is detected, with active TRIP condition or test is inhibited by a user-configured enabling signal)
- Setpoints can be varied in a user-defined range to prevent deposits in the valve housing
- The valve feedback can be monitored as early as the startup phase of the Partial Stroke Test so that early valve failure is detected
- PFD calculations to forecast the next required Full Stroke Test.

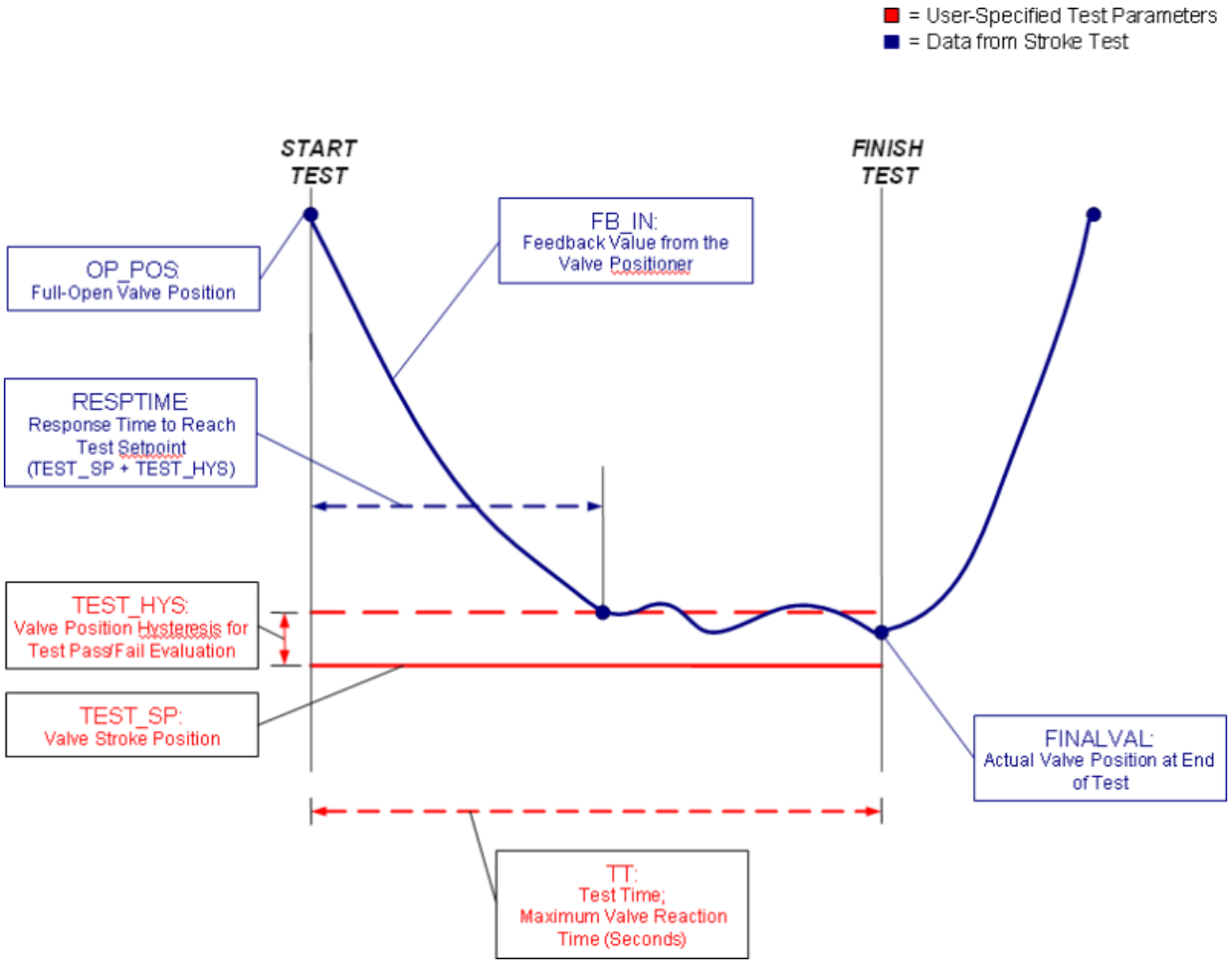
The following figures illustrate the basic functionality of the Partial Stroke Test logic. Before running a test, the user specifies the setpoint for the valve test (`TestSP`), a test hysteresis (`TestHyst`) and the maximum test running time (`TT`). When a new test is started, the setpoint of the valve position controller is set to `TestSP`, a timer is started and the feedback for the actual valve position (`F_PST.FB_IN`) is monitored. The test is passed if `F_PST.FB_IN` reaches a value below the pass threshold value. This is calculated as follows: `TestSP` + `TestHyst`. If `F_PST.FB_IN` has not passed through the success threshold value within the time interval predefined as the maximum test running time, the test has failed. At the end of a (passed or failed) test, the valve returns to its fully open position.

If a passed Partial Stroke Test is detected, the time that the valve required to pass through the threshold value `TestSP` + `TestHyst` is shown as response time (`RespTime`). The test is stopped if the value of the valve feedback is either less than `TestSP` (for display, see "Function sequence for the Partial Stroke Test (feedback value reached `TestSP`)") or if the time `TT` has passed (for display, see "Function sequence for Partial Stroke Test (feedback value reached `TestSP` + `TestHyst`)" – depending on what occurs first. At this point, `F_PST.FB_IN` is displayed and saved as the actual valve position at the end of the test (`FinalVal`). The values for `RespTime` and `FinalVal` are saved as data for prospective maintenance.

Function sequence of the Partial Stroke Test (feedback value reaches `TestSP`):




Function sequence of the Partial Stroke Test (feedback value reaches `TestSP + TestHyst`):




### 4.5.4.2 Logic for the solenoid valve test (optional)

If pneumatically actuated valves are used, optionally available Partial Stroke Test logic can be used for a solenoid valve test. Solenoid valve test is only relevant to hardware configurations with a solenoid valve.

The following figure shows the basic functional sequence of the logic for the solenoid valve test. If the solenoid valve test has been enabled in the configuration, it is automatically performed after a Partial Stroke Test has been passed. A delay configurable by the user occurs between the end of the Partial Stroke Test and the start of the solenoid valve test, so that the process can stabilize before the solenoid valve test. The logic for the solenoid valve test sends a fail-safe, discrete output signal as a pulse to the solenoid valve (and thereby sets it from TRUE to FALSE and then back to TRUE). The logic then monitors the feedback of the valve position controller for valve movement. For the solenoid valve test to be passed, a pre-configured stroke value has to be met within a pre-configured time interval. This stroke value cannot be a greater value, as the Partial Stroke Test that has just been passed has already confirmed that the valve has free movement. The test now checks whether the solenoid valve is fulfilling its function of interrupting the compressed air supply.

When setting up the test, the user specifies the following parameters:

- `DT_S`: Delay time between end of a passed Partial Stroke Test and start of the solenoid valve test
- `TT_S`: The test time is equal to the number of cycles for which the fail-safe output to the solenoid valve is set to 'off' by a pulse
- `MonTi_S`: The time interval within which the logic monitors the valve for a stroke movement
- `MonVal_S`: The stroke value that must be identified by the valve position controller for the solenoid valve test to be passed

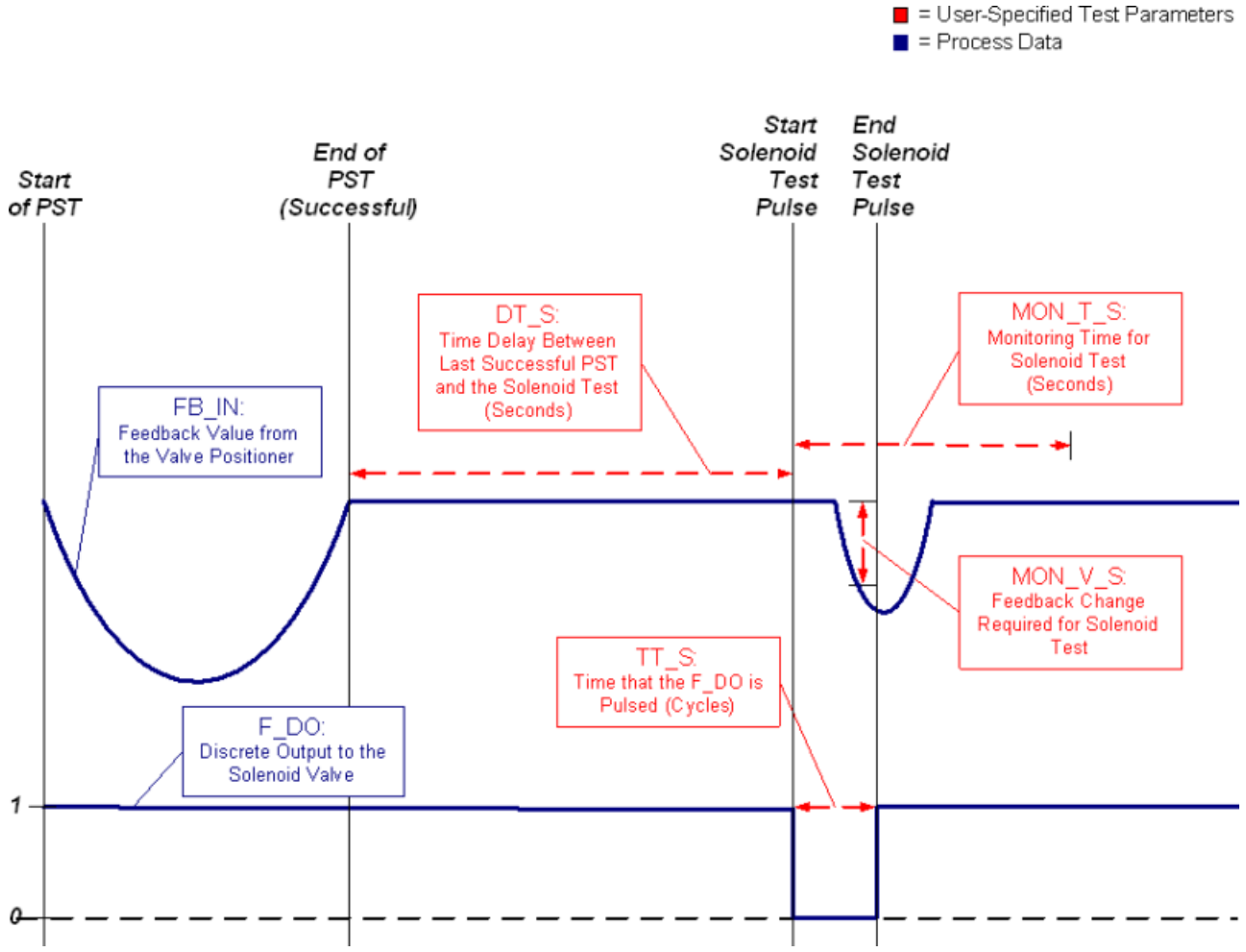
The logic of the solenoid valve test is only used in certain cases. Firstly, it is only suitable for hardware configurations with solenoid valve. Whether a solenoid valve test is meaningful depends on the cycle time of the application and the response time of the valve. In particular with regard to fast-response valves, the test pulse can cause a process fault. The user is responsible for determining whether the applicable solenoid valve responds too quickly to be tested by the test logic. Prior to using the logic for the solenoid valve test, the PFD values of the solenoid valve should be checked. If the solenoid valve makes no significant contribution to the PFD of the overall valve body, it may not be necessary to test the solenoid valve regularly with the logic of the solenoid valve test.

**Note**

When defining the test running time (`TT_S`), no time interval may be chosen in which the solenoid valve test will close the valve too far and therefore cause false tripping. `TT_S` should be significantly shorter than the response time of the solenoid valve.

The logic of the solenoid valve test must not be used for fast-response valve A response of the valve that is too fast can cause false tripping.

Functional sequence of the solenoid valve test:




### 4.5.4.3 Adding the Partial Stroke Test engineering template to a STEP 7 project

#### (1) Copy template to a CFC chart

The template can be copied to a user project in two ways.

- Copy template to the SIMATIC Manager:
  - Open the Condition Monitoring Library in the SIMATIC Manager
  - Copy the PST_Template for CFC charts into the project folder for CFC charts.
- Copy template to the CFC chart:
  - Open a CFC chart in the user project and select the library catalog. The Condition Monitoring Library is available here.
  - Select "Condition Monitoring Library" and then select "S7 program\Charts".
  - Drag the PST_Template for CFC charts to the open CFC chart.

In both cases, all blocks required to run the Partial Stroke Test application in the PLC are automatically copied to the user project.

**Note**

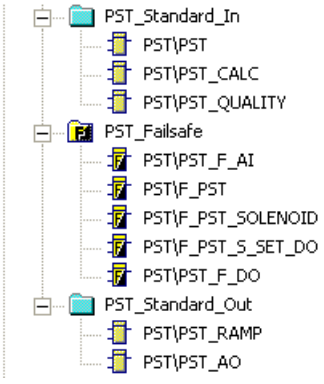
All the blocks in the Partial Stroke Test template must be run in the pre-configured sequence. This ensures that the Partial Stroke Test application logic (in terms of points in time and run sequence) functions as intended.

### (2) Set up run sequence

When the template is copied to the CFC chart, all the blocks are copied to the same runtime group. As the fail-safe program must be run in a separate runtime group, there are three runtime groups.

Open the runtime editor in the CFC chart and order the blocks as shown in the following figure.

Run sequence of the blocks in the Partial Stroke Test template:



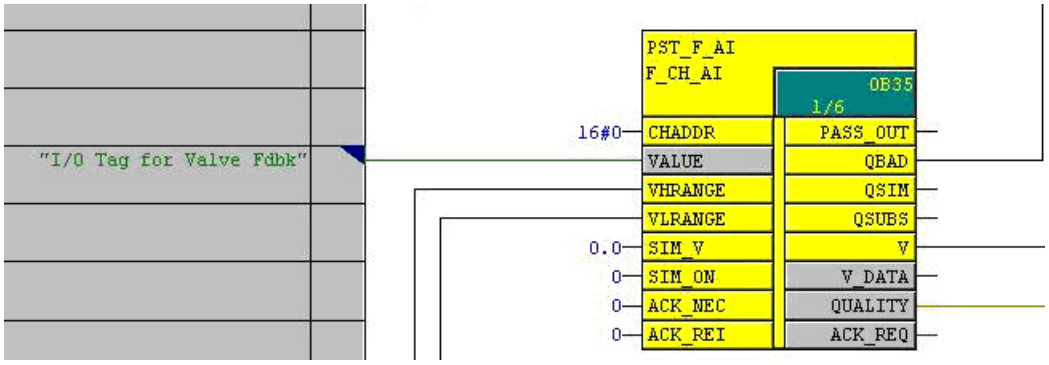

### (3) Adjust the connections to the I/O channel drivers

The Partial Stroke Test engineering template contains three channel driver blocks:

- One F_CH_AI channel driver block for the feedback of the valve position controller
- One Pcs7AnOu block for the setpoint of the valve position controller
- The template contains a F_CH_DO channel driver block that is used to control the solenoid valve. If the Safety Instrumented Function and the corresponding channel driver block are not available, the F_CH_DO block is not required.

The instance name of the F_CH_AI block in the template is PST_F_AI. The PST_F_AI.VALUE input of the channel driver block must be connected to the symbolic address of the channel for the feedback of the valve position controller (see following figure). Note that this analogue input channel must initially be created in the hardware configuration.

F_CH_AI channel driver block in the Partial Stroke Test template:




The Partial Stroke Test application is designed to evaluate signals from a fail-safe analog input for the feedback of the position controller. The input for the feedback, the associated test evaluation logic and the timers for the test intervals are created with fail-safe logic to ensure that the valve stroke occurs within the configured Partial Stroke Test interval. The fail-safe logic ensures that no undiscovered error (e.g. a transient RAM error or a communications error) incorrectly results in a successful Partial Stroke Test. If this level of safety is not required for an application, the signal from a standard analog input can be used instead of a fail-safe analog input.

If a standard analog input signal or a PROFIBUS PA analog input signal is used, the F_CH_AI channel driver block must be deleted and replaced by a Pcs7AnIn channel driver block (for a standard analog input) or FbAnIn (for a PROFIBUS PA analog input). To do so, the user proceeds as follows:

- Delete F_CH_AI block from the CFC chart
- Add a Pcs7AnIn / FbAnIn driver block to the CFC chart and rename to PST_AI (as per the existing naming structure)
- Connect output `PST_AI.PV_Out` (Pcs7AnIn) or `PST_AI.PV_Li` (FbAnIn) to input `PST_QUALITY.IN1`
- Add a F_R_FR conversion block to the CFC chart and rename to F_PST_CONV
- Connect output `PST_AI.PV_Out` (Pcs7AnIn) or `PST_AI.PV_Li` (FbAnIn) to input `F_PST_CONV.IN`
- Connect output `F_PST_CONV.OUT` to input `F_PST.FB_IN`
- Check and if necessary change the assignment of the execution group

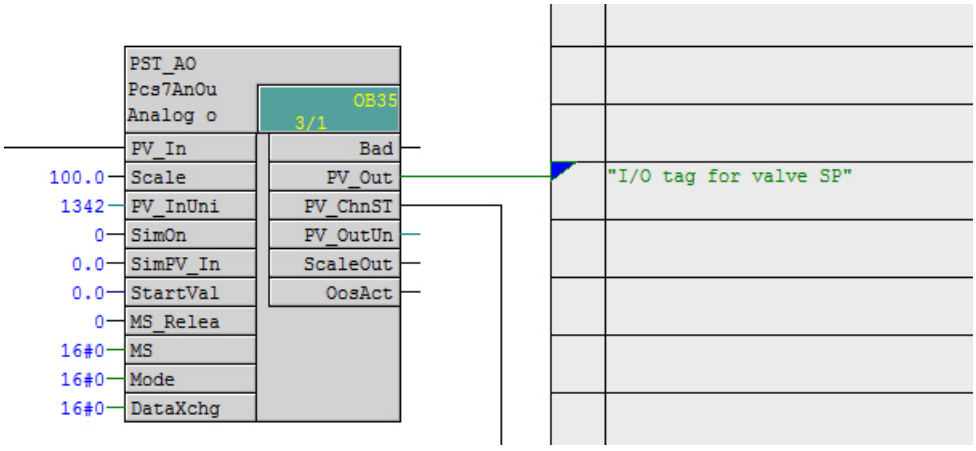
The instance name of the Pcs7AnOu block in the template is PST_AO. The `PST_AO.PV_Out` output of the channel driver block must be connected to the symbolic address of the channel for the setpoint of the valve position controller (see figure below). Note that this analogue output channel must initially be created in the hardware configuration. The signal does not have to be fail-safe, as the feedback signal already correctly indicates whether the test was run within the test interval.

If a PROFIBUS PA analog output is used, the Pcs7AnOu channel driver block must be deleted and replaced by a FbAnOu channel driver block. To do so, the user proceeds as follows:

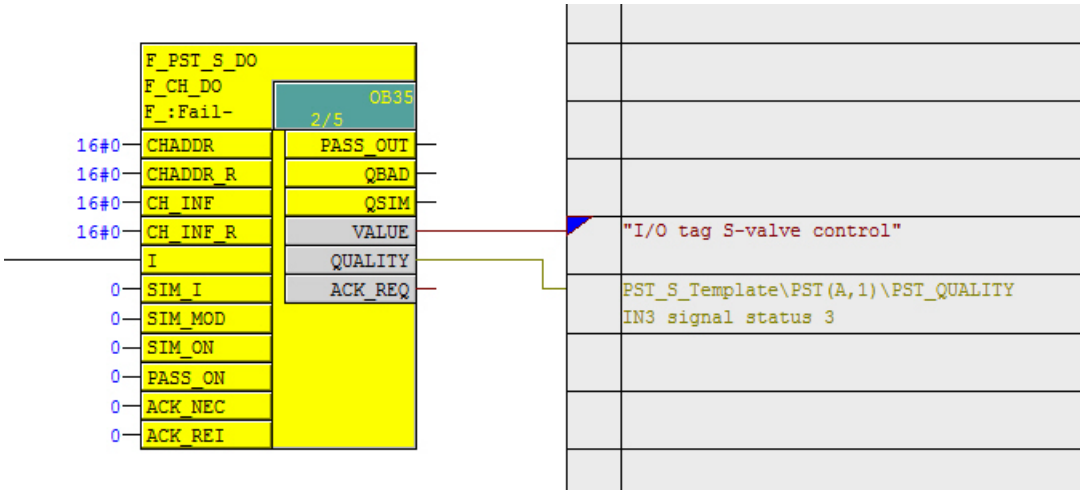
- Delete the Pcs7AnOu block from the CFC chart
- Add a FbAnOu driver block to the CFC chart and rename to PST_AO (as per the existing naming structure)
- Connect output `PST_AO.PV_ChnST` to input `PST_QUALITY.IN2`

- Connect output `PST.SP` to input `PST_AO.SP_Li`
- Check and if necessary change the assignment of the execution group

Pcs7AnOu channel driver block in the Partial Stroke Test template:




F_CH_DO channel driver block in the Partial Stroke Test template:




### 4.5.4.4 Configure blocks

Information on how to configure the individual blocks is provided in the applicable descriptions of the blocks in the "Configuration" section.

- AUTOHOTSPOT
- AUTOHOTSPOT
- Description of PST_Calc (Page 657)

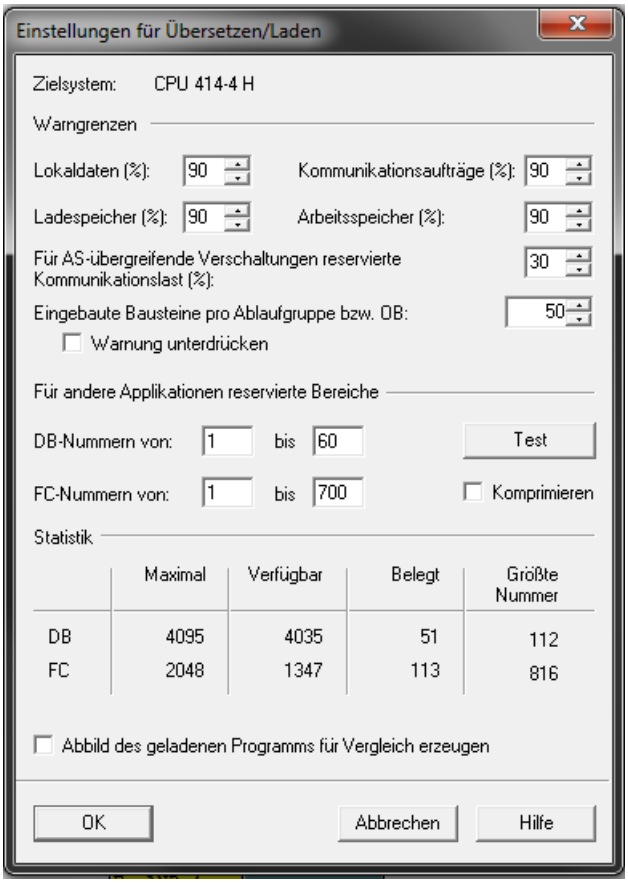
- Description of PST_OrQC (Page 662)
- Description of PST_Time (Page 664)

#### 4.5.4.5 Compile and load logic of the Partial Stroke Test

After the Partial Stroke Test template has been added to a CFC chart and configured (i.e. all necessary I/O connections have been made and the test parameters F_PST, F_PST_S, PST and PST_Calc have been defined), the logic can be compiled. After compiling, the logic can be loaded to the controller.

Two notes on compiling the logic:

- The Partial Stroke Test engineering template uses STEP 7 functions (FCs) in a low range (i.e. FC1, "AD_DT_TM", FC2 "CONCAT", FC16 "I_STRNG", FC32 "RIGHT" etc.) by default. The number of the FCs must not be changed. For the logic to compile successfully, the user must allow FCs above number 1 to be taken into account on compile.
  To do so, proceed as following in the CFC editor: Select "Extras > Settings > Compile/Load" to open the "Settings for Compile/Load" dialog. In the "FC numbers from:" field, enter "1" for the start of the range and close the dialog with "OK" (see figure below). Then compile the logic as normal.
  Compile/load dialog in the CFC editor:




- The Partial Stroke Test engineering template contains multiple I/O channel driver blocks: For the application to function as intended, specified I/O diagnostic blocks must be created automatically and connected during compile.
  To create I/O diagnostic blocks, select "Chart > Compile > Chart as program" in the CFC editor. The "Compile program" dialog opens. Enable the "Create module driver" option and continue to compile the logic.

## 4.5.5 Working with the engineering template

### 4.5.5.1 Working with the engineering template

The engineering template can be operated and monitored in runtime mode. Open the Partial Stroke Test template in the CFC editor to view the logic details (individual blocks such as F_PST and the Partial Stroke Test block must be visible). Real-time data can be viewed in the "Debug" menu via the "Test mode" and "Monitoring on" > "Register I/Os for monitoring" menu options.

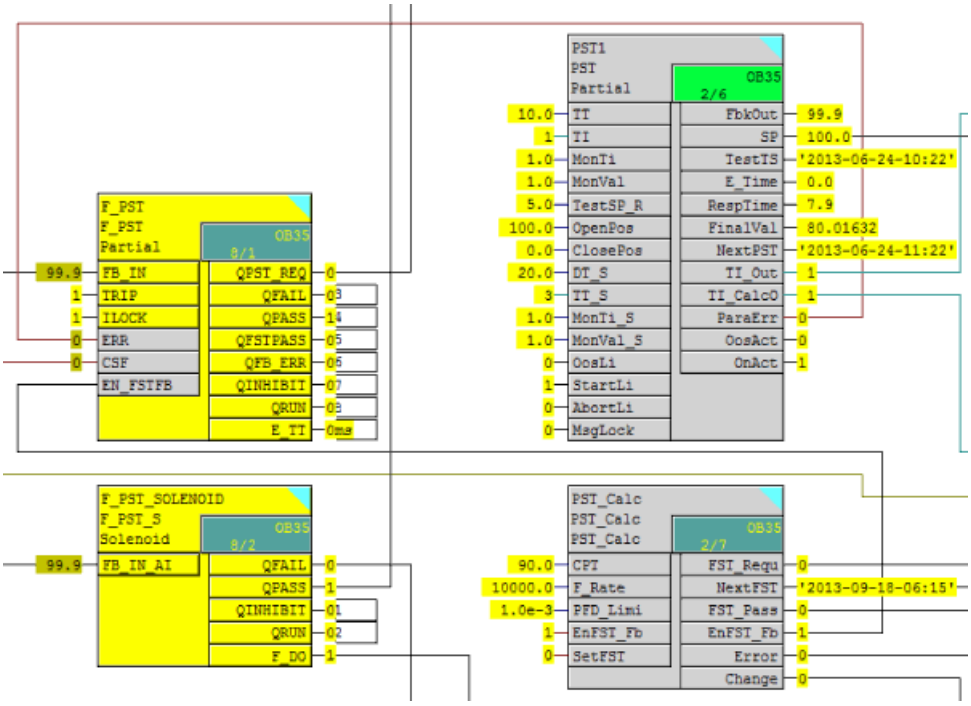
### 4.5.5.2 Display Partial Stroke Test data

Real-time data can be monitored after switching the Partial Stroke Test logic to runtime mode. The following figure shows the Partial Stroke Test logic running in monitoring mode with real-time data for the Partial Stroke Test visible. In this case, the data includes the following:

- `F_PST.QPASS = 1`; indicates that the Partial Stroke Test was passed
- `PST.TestTS = 2013-06-24-10:22`; indicates that the last Partial Stroke Test was run on June 24, 2013 at 10:22
- `PST.RespTime = 7.9`; indicates that the time for the valve position to pass through the threshold (`TestSP` + `TestHyst`) for the last Partial Stroke Test was almost 8 seconds
- `PST.FinalVal = 80.01632`; indicates that the last Partial Stroke Test ended at valve position 80.0

- `PST.NextPST = 2013-06-24-11:22`; indicates that the next Partial Stroke Test is planned for June 24, 2013 at 11:22
- `PST_Calc.NextFST = 2013-09-18-06:15`; indicates that the next Full Stroke Test is required on September 18, 2013 at 6:15 to meet the specified PFD requirements.

Partial Stroke Test logic in runtime mode:




### 4.5.5.3 Changing the values of Partial Stroke Test parameters

Any test parameters may be modified when the Partial Stroke Test logic is in runtime mode. However, subsequent changes are normally only necessary for the following Partial Stroke Test parameters:

- `TI`, interval of the automatic test
- `TestSP`, Partial Stroke Test setpoint
- `TestHyst`, hysteresis of the valve position for the success/failure result
- `TT`, Partial Stroke Test time
- `MonTi`, monitoring time for the start of the Partial Stroke Test
- `MonVal`, required change to the feedback within the time period `MonTi`

**Note**

If the value for `TI` is changed while a timer is running, the timer is fully reset. The test is then only run when the time interval of the new `TI` value is fully completed. In this case, we recommended that the user performs a manual Partial Stroke Test to cover the time interval before `TI` changes.

**Note**

If the valve closes for some reason (e.g. emergency shutdown, planned maintenance etc.) below the threshold (`ClosePos + TestHyst`), this is then only a Full Stroke Test if `PST_Calc.EnFST_Fbk = 1`. All times are reset. The PST_Calc block sets the time to the time of the next Full Stroke Test.

Otherwise, i.e. at `PST_Calc.SetFST = 1`, a Full Stroke Test is registered. Similarly in this case, all timers are reset and the PST_Calc block sets the time to the time of the next Full Stroke Test.

#### 4.5.5.4 Changing the values of solenoid valve test parameters

Normally, the following solenoid valve test parameters have to be set subsequently:

- `DT_S`, delay time for the solenoid valve test after the Partial Stroke Test has completed
- `TT_S`, time period for the solenoid valve test
- `MonTi_S`, monitoring time for the solenoid valve test
- `MonVal_S`, required change to the feedback within the time period `MonTi_S`

#### 4.5.5.5 Starting or aborting the Partial Stroke Test manually

When the Partial Stroke Test logic is in runtime mode, a Partial Stroke Test can be started or a running test can be aborted.

- To start a test, there must be a pulse for the change from 0 to 1 to input `StartLi` of the PST block. As a test can only be started when `StartLi` changes from 0 to 1, `StartLi` must be reset to 0 before a test can be requested manually.
  It is also possible to start the Partial Stroke Test from the faceplate. The I/O parameter `StartOp` is set by the faceplate and reset by the PST block after readout.
- To abort a running test, there must be a pulse for the change from 0 to 1 to input `AbortLi` of the PST block. As a test can only be aborted when `AbortLi` changes from 0 to 1, `AbortLi` must be reset to 0 before a test can be requested manually.
  It is also possible to abort the Partial Stroke Test from the faceplate. The I/O parameter `AbortOp` is set by the faceplate and reset by the PST block after readout.

### 4.5.6 F_PST

**Note**

This block is available as a part of PST template can be downloaded from the SIEMENS industry Online Support page. You can find the link in the APL V9.1 Readme (Online).

## 4.5.7 F_PST_S

**Note**

This block is available as a part of PST template can be downloaded from the SIEMENS industry Online Support page. You can find the link in the APL V9.1 Readme (Online).

## 4.5.8 PST_Calc block

### 4.5.8.1 Description of PST_Calc

#### Object name (type + number) and family

Type + number: FB 1957

Family: CondMon

Number of process objects: 0 PO

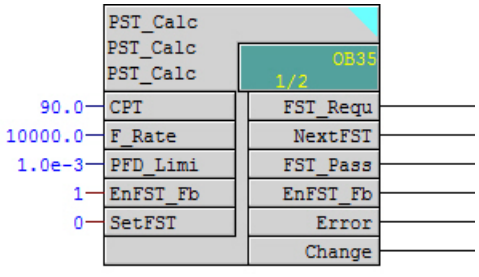
#### Mode of operation

The PST_Calc block contains logic that calculates when the next full proof test of the valve body is required (see following figure). The Partial Stroke Test application defines this time as the date for the Full Stroke Test, but the user must specify the scope of the proof test that is required to test the entire valve body.

The block uses the data for the rate of occurrence of failure, the value for the Probability of Failure on Demand (PFDAVG) and the effect of the Partial Stroke Test to calculate the test interval to the next test required for the desired SIL level. The PST_Calc logic is responsible for the following tasks:

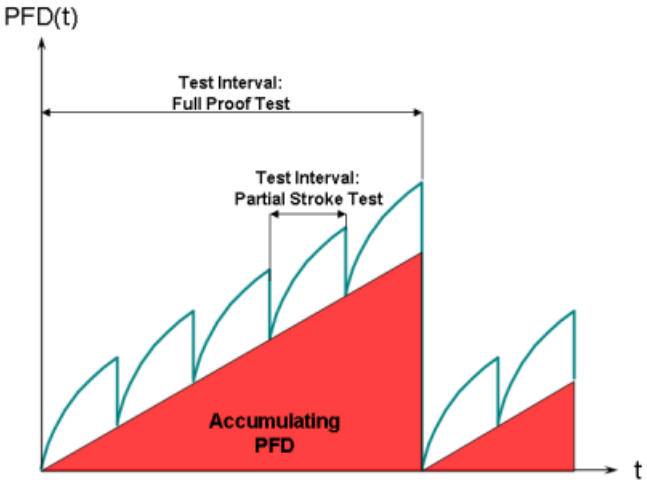
- Calculation of the running PFDAVG for the valve
- Output of a message for the user stating the next required Full Stroke Test (via PST block)
- Display of date and time at which the next Full Stroke Test is required

PST_Calc block in the valve startup test template:



A safety shutoff valve is not fully closed in a Partial Stroke Test. Not all failure modes of the valve can therefore be tested with this online test method. As already noted, the diagnostic coverage of an online proof test (CPT) is used a percentage measures of the effectiveness of a Partial Stroke Test. If the CPT value for a safety shutoff valve is 70%, for example, a Partial Stroke Test can detect 70% of all possible failure modes of the valve. Conversely, this means that 30 % of all the valve's possible failure modes cannot be tested. As these potentially undiagnosed errors accumulate, the Probability of Failure on Demand (PFDAVG) of the valve increases over time. The value for PFDAVG can only be reset to zero by a complete proof test (Full Stroke Test and associated tests), as this is the only way to achieve diagnostic coverage of 100 %. The following figure illustrates this concept:

Cumulation effect for the PFD value with the Partial Stroke Test:

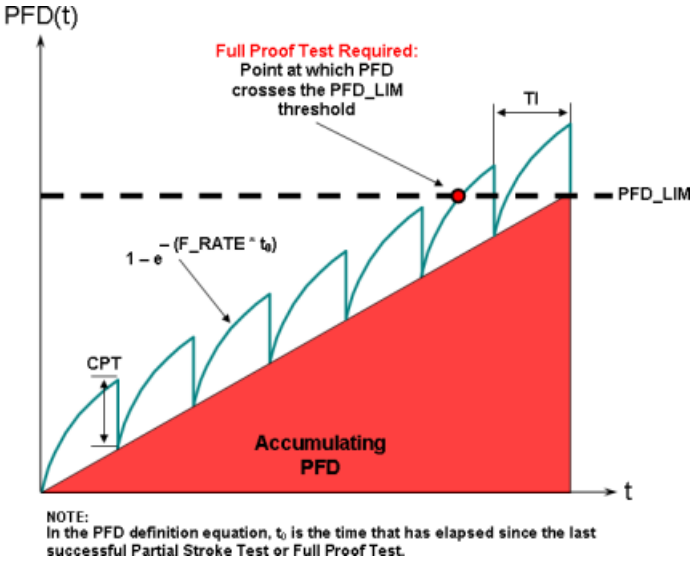



After automatic tests are enabled for the first time, the PST_Calc block performs a series of calculations to determine the time of the next required full proof test (Full Stroke Test). PST_Calc calculates the curve for PFDAVG on the basis of criteria entered by the user. The PFDAVG curve for the valve is then updated when a new Partial Stroke Test or Full Stroke Test is run. This includes manually running these tests, which is also detected by the logic via monitoring of the feedback for the valve position. The PST_Calc block requires four values to calculate the next test date and time successfully:

- `TI`: The time interval between automatic Partial Stroke Tests (this input value for the PST block is transferred to the PST_Calc block)
- `CPT`: The Coverage of a Proof Test for the Partial Stroke Test
- `F_Rate`: The rate of dangerous undetected failures (λDU) for the valve
- `PFD_Limit`: The threshold value that must be met to comply with the desired SIL. If the calculated PFD value reaches the threshold value, a full proof test (Full Stroke Test) is required for the valve. This value is provided for the desired SIL in the SIL tables of the IEC 61511 standard.

The following figures illustrates the relationship between the parameters for PST_Calc.

Principle of operation of PST_Calc:




## Configuration

The inputs and outputs of the PST block that have to be parameterized by the user are described in the following two tables.

### Inputs of the PST_Calc block

| Parameter | Description | Type | Default |
|---|---|---|---|
| CPT | Effectiveness of the online proof test in percent (0 % - 100 %)<br>CPT should be set to the percentage value of the failure modes that a Partial Stroke Test covers with the test setpoint (TestSP). | REAL | 0.0 |
| F_Rate | Valve failure in time rate (FIT)<br>F_Rate defines the rate of dangerous undetected failures (λDU) for the valve body. | REAL | 100.0 |
| PFD_Limit | PFD threshold per SIL<br>PFD_Limit specifies the threshold value for PFDAVG below which the valve body must remain to achieve the desired SIL for the Safety Instrumented Function.<br>The PFDAVG values required for specified SIL levels are provided in the standard IEC 61511. | REAL | 100.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| EnFST_Fbk | Enabled to record the Full Stroke Test feedback (1=enabled)<br>EnFST_Fbk allows the logic to register a Full Stroke Test solely from the valve position controller's feedback.<br>If enabled (with EnFST_Fbk=1), the logic always then registers a Full Stroke Test when the feedback indicates that the valve is fully closed (F_PST.FB_IN < (PST.ClosePos + PST.TestHyst)).<br>If disabled (with EnFST_Fbk= 0), no Full Stroke Test is registered under the same preconditions. In this case, the user can specify via the input SetFST that a Full Stroke Test has taken place. | BOOL | 1 |
| SetFST | Manual report of a Full Stroke Test (1=New Full Stroke Test)<br>When SetFST switches from 0 to 1, a new Full Stroke Test is registered. This resets the calculation of PFDAVG.<br>SetFST can be connected to the user-configured logic. As a new Full Stroke Test is only registered when SetFST switches from 0 to 1, the input has be to reset between the Full Stroke Tests. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0 u<br>• 16#80 |

## Outputs of the PST_Calc block

| Parameter | Description | Type | Default |
|---|---|---|---|
| FST_Request | Full Stroke Test required (1=Warning active)<br>FST_Request specifies the time at which a Full Stroke Test is required to maintain the desired SIL for the Safety Instrumented Function. This value is set to 1 if the time (date/time) specified by NextFST is exceeded at the current time. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NextFST | Time stamp for next Full Stroke Test<br>NextFST outputs a text string that specifies the time of the next required Full Stroke Test.<br>The Full Stroke Test must be performed at the specified time to maintain the desired SIL for the Safety Instrumented Function. | Data type: STRING [16] | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Error | Indicates invalid parameters (1=Invalid parameters)<br>The following validity checks are performed for the PST_Calc block:<br>• CPT >= 0 and CPT < 100<br>• F_Rate > 0<br>If an invalid parameter is detected, Error is set t o1.<br>PST_Calc is only calculated if automatic tests are first enabled (e.g. for PST.TI > 0) and then a Full Stroke Test is performed.<br>To reset Error and force the calculation to update, a Full Stroke Test must be performed or automatic tests must be disabled and re-enabled. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Change | Indicates parameter change (1=Parameter changed)<br>If one of the following parameter values changes between the two Full Stroke Test (where automatic tests are enabled, the block calculations are invalid and Change is set to 1:<br>• PST.TI<br>• CPT<br>• F_Rate<br>• PFD_Limit<br>PST_Calc is only calculated if automatic tests are first enabled (e.g. for PST.TI > 0) and then a Full Stroke Test is performed.<br>To reset Change and force the calculation to update, a Full Stroke Test must be performed or automatic test must be disabled and re-enabled. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

**Note**

The PST_Calc block has additional inputs and outputs that are not documented in the two above tables. These inputs and outputs are used by the Partial Stroke Test logic to transfer data between blocks and are normally hidden. The user should only define the parameters described in the table.

### 4.5.8.2 I/Os of PST_Calc

#### I/Os of PST_Calc

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

#### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| CPT | Diagnostic coverage of the Partial Stroke Test [%] | REAL | 0.0 |
| EnFST_Fbk | Evaluation of the Full Stroke Test on the basis of the feedback | BOOL | 1 |
| f_FST_Passed | New passed Full Stroke Test [feedback] | UDT_F_B | |
| f_Passed | New passed Partial Stroke Test | UDT_F_B | |
| F_Rate | Valve failure in time rate [FIT] | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PFD_Limit | PFD threshold value as per required SIL | REAL | 0.0 |
| SetFST | New passed Full Stroke Test (manual input) | BOOL | 0 |
| TI | Test interval [hours] | INT | 0 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Change | 1=Parameter has been changed | DigVal | |
| EnFST_FbkOut | Evaluation of the Full Stroke Test on the basis of the feedback | DigVal | |
| Error | Error in internal block | DigVal | |
| FST_Passed | Evaluation of the inputs f_FST_Passed and SetFST | DigVal | |
| FST_Request | Full Stroke Test required (1=Warning active) | DigVal | |
| NextFST | Time and date for next Full Stroke Test | STRING[16] | |

## 4.5.9 PST_OrQC block

### 4.5.9.1 Description of PST_OrQC

### Object name (type + number) and family

Type + number: FC 449

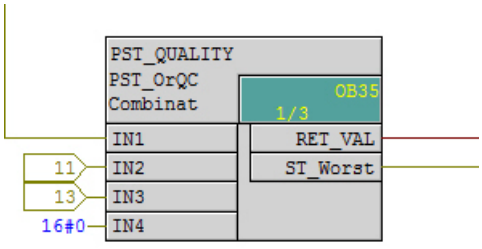
Family: CondMon

Number of process objects: 0 PO

### Mode of operation

The PST_OrQC block contains logic that evaluates the quality codes of a maximum of four driver blocks for possible poor signal quality. The block returns a single output value. The block evaluates the quality codes of the driver blocks that are used in the engineering template for the Partial Stroke Test. The result can then be connected to the CSF input of the F_PST block.

PST_OrQC block in the valve startup test template:



### Configuration

The inputs and outputs of the PST block that have to be parameterized by the user are described in the following two tables.

### Inputs of the PST_OrQC block

| Parameter | Description | Type | Default |
|---|---|---|---|
| INx | Input x (x=1..4)<br>If IN1, IN2, IN3 and IN4 are equal to 16#80, the quality of the signal is good.<br>If one of the inputs INx is not equal to 16#80, the quality of the signal is bad. | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### Outputs of the PST_OrQC block

| Parameter | Description | Type | Default |
|---|---|---|---|
| RET_VAL | Return value of the function<br>If all the input signals return a quality code for "good", RET_VAL is set to FALSE.<br>If at least one of the inputs returns a quality code for "bad", RET_VAL is set to TRUE. | BOOL | |

## 4.5.9.2 I/Os of PST_OrQC

### I/Os of PST_OrQC

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| IN1 | Signal Status 1 | AnaVal | 16#80 |
| IN2 | Signal Status 2 | AnaVal | |
| IN3 | Signal Status 3 | AnaVal | |
| IN4 | Signal Status 4 | AnaVal | |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| RET_VAL | Return value of the function | | |
| ST_Worst | Worst signal status | BYTE | 16#80 |

## 4.5.10 PST_Time block

### 4.5.10.1 Description of PST_Time

#### Object name (type + number) and family

Type + number: FC 448

Family: CondMon

Number of process objects: 0 PO

#### Mode of operation

The PST_Time block contains logic that adds a time constant to the time stamp (of a passed test) and returns the value as a character string for display purposes.

#### Configuration

The PST_Time block is called by the PST block (within the SCL code of the PST block). The block is not visible to the user in the Partial Stroke Test engineering template, but belongs to the Partial Stroke Test application and the Condition Monitoring Library.

### 4.5.10.2 I/Os of PST_Time

#### I/Os of PST_Time

For the parameters that are not linkable, refer:Parameters that cannot be connected (Page 75)

#### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `ActTime` | Actual time<br>`ActTime` is the original time stamp. | DATE_AND_TIME | DT#90-1-1-0:0:0.000 |
| `OffsetH` | Time delay (hours)<br>`OffsetH` is the time delay in hours that is added to `ActTime`. | REAL | 0.0 |
| `OffsetMs` | Time delay (milliseconds)<br>`OffsetMs` is the time delay in milliseconds that is added to `ActTime`. | TIME | T#0s |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| RET_VAL | Return value of the function<br>RET_VAL is the sum of ActTime, OffsetH and OffsetMs after conversion to a format for character strings. | STRING[16] | |

## 4.5.11 PST Block

### 4.5.11.1 Description of PST

#### Object name (type + number) and family

Type + number: FB 1956

Family: CondMon

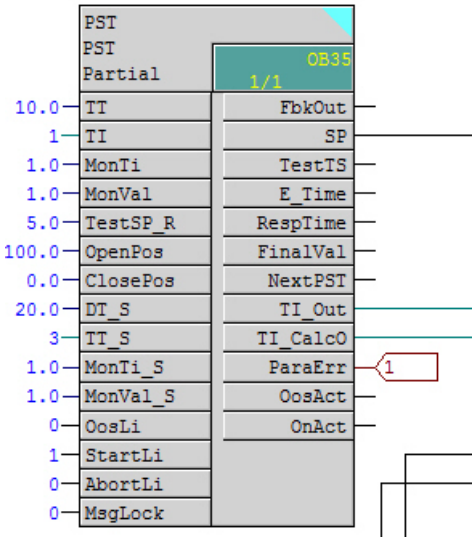
Number of process objects: 30 PO

#### Mode of operation

The PST block contains the standard logic for the Partial Stroke Test application (see following figure). The PST logic is responsible for the following tasks:

- Acts as interface to the block icon for the Partial Stroke Test/solenoid valve test (for data transfer to/from the PCS 7 OS)
- Records the time the valve requires to reach the desired stroke setpoint during a Partial Stroke Test
- Records the feedback of the valve position controller at the end of a Partial Stroke Test
- Controls the valve position controller setpoint
- Defines the messages for Partial Stroke Test/solenoid valve test (to be used together with PCS 7 OS)

PST block in the Partial Stroke Test template:




## Configuration

The inputs and outputs of the PST block that have to be parameterized by the user are described in the I/O descriptions (AUTOHOTSPOT).

### Startup characteristics

Use `Feature.Bit0` (Setting the startup characteristics) to define the startup characteristics of this block.

- **Bit = 0:** The most recently stored values are reset at startup.
- **Bit = 1:** The most recently used values are sent to the following output parameters at startup:
  - `SP_UpRaAct`
  - `SP_DnRaAct`
  - `TripAct`
  - `PST_ReqAct`
  - `FailedAct`
  - `PassedAct`
  - `FST_PassedAct`
  - `FbkErrActAct`
  - `InhibitAct`
  - `AbortedAct`
  - `RunAct`
  - `Failed_S_Act`
  - `Passed_S_Act`
  - `Inhibit_S_Act`
  - `Run_S_Act`
  - `StartEnAct`
  - `StartEn_S_Act`

**Note**

Additional information about `Feature.Bit0` can be found in the *PCS 7 Advanced Process Library* manual.

### Status word assignment for the `Status1` parameter

For a description of the individual parameters, refer to the section AUTOHOTSPOT.

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `Not used` |
| 3 | `"Out of service" operating mode (OosAct.Value)` |
| 4 | `"Out of service" operating mode (OosLi.Value)` |
| 5 | `OnAct.Value` |

| Status bit | Parameter |
|---|---|
| 6 | MsgLock.Value |
| 7 | Started_MsgEn |
| 8 | Ended_MsgEn |
| 9 | Passed_MsgEn |
| 10 | Failed_MsgEn |
| 11 | Aborted_MsgEn |
| 12 | FbkErr_MsgEn |
| 13 | Inhibit_MsgEn |
| 14 | FST_Passed_MsgEn |
| 15 | FST_Request_MsgEn |
| 16 | PST_Request_MsgEn |
| 17 | ErrCalc_MsgEn |
| 18 | ChangeCalc_MsgEn |
| 19 | Started_S_MsgEn |
| 20 | Ended_S_MsgEn |
| 21 | Passed_S_MsgEn |
| 22 | 1 = setpoint ramp active |
| 23 | Inhibit_S_MsgEn |
| 26 | 1 = automatic test active |
| 27 | 1 = ParaErr parameter error |
| 28 | Internal Error |
| 29 | 1= Suppressed messages |
| 30 - 31 | Not used |

### Status word assignment for `Status2` parameter

For a description of the individual parameters, refer to the section AUTOHOTSPOT.

| Status bit | Parameter |
|---|---|
| 0 | Partial Stroke Test started |
| 1 | Partial Stroke Test completed |
| 2 | Not used |
| 3 | Partial Stroke Test passed |
| 4 | Partial Stroke Test failed |
| 5 | Partial Stroke Test failed because valve is blocked |
| 6 | Partial Stroke Test aborted |
| 7 | Partial Stroke Test feedback error |
| 8 | Partial Stroke Test NOT performed because trigger state not met |
| 9 | Partial Stroke Test NOT performed because enable precondition not met |
| 10 | Partial Stroke Test NOT performed because feedback is poor quality |
| 11 | Partial Stroke Test NOT performed because feedback below tolerance |
| 12 | Partial Stroke Test NOT performed because solenoid valve test is running |
| 13 | Partial Stroke Test NOT performed because parameter invalid |

| Status bit | Parameter |
|---|---|
| 14 | Full Stroke Test passed |
| 15 | Full Stroke Test required |
| 16 | Partial Stroke Test required |
| 17 | Calculation parameter assignment error |
| 18 | Calculation parameter change |
| 19 | Solenoid valve test started |
| 20 | Solenoid valve test completed |
| 21 | Solenoid valve test passed |
| 22 | Solenoid valve test failed |
| 23 | Solenoid valve test NOT performed |
| 24 | Solenoid valve test running |
| 25 | Partial Stroke Test running |
| 26 | FailedAct |
| 27 | InhibitAct |
| 28 | Solenoid valve test enabled |
| 29 | TripAct |
| 30 | ILockAct |
| 31 | Not used |

### 4.5.11.2 PST operating modes

This block can be set to the following modes:

- On
- Out of service

#### "On"

General information about the "On" operating mode can be found in the "On" section of the *"PCS 7 Advanced Process Library"* manual.

#### "Out of service"

General information about the "Out of service" operating mode can be found in the "Out of service" section of the *PCS 7 Advanced Process Library* manual.

### 4.5.11.3 PST functions

The functions for this block are listed below.

#### Forming the signal status for blocks

This block provides the standard function "Forming and outputting the signal status for technologic blocks" (see the *"PCS 7 Advanced Process Library"* manual).

The worst signal status `ST_Worst` for the block is formed from the following parameter `ST_WorstIn`. This is connected to the PST_OrQC block.

### Configurable reactions using the `Feature` parameter

An overview of all the reactions that are provided by the `Feature` parameter is available in section "Configurable response using the Feature I/O" (see the *PCS 7 Advanced Process Library* manual). The following modes of behavior are available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Setting the startup characteristics |
| 1 | Characteristics for the out of service mode |
| 22 | Update acknowledgment and error status of the alarm call |
| 24 | Enabling local operator authorization |

### Operator control permissions

This block provides the standard function "Operator control permissions" (see the *"PCS 7 Advanced Process Library"* manual).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | 1 = Operator can switch to "Out of service" mode |
| 3 – 9 | Not used |
| 10 | 1 = User can start a manual Partial Stroke Test |
| 11 | 1 = User can abort a manual Partial Stroke Test |
| 12 | 1 = User can change the automatic test interval for the Partial Stroke Test |
| 13 | 1 = User can change the test setpoint for the Partial Stroke Test |
| 14 | 1 = User can change the position tolerance for the Partial Stroke Test |
| 15 | 1 = User can change the maximum test time for the Partial Stroke Test |
| 16 | 1 = User can change the delay time for the solenoid valve test |
| 17 | 1 = User can change the minimum position change for the solenoid valve test |
| 18 – 31 | Not used |

### Gradient limit of the setpoint

This block provides the standard function "Gradient limit of the setpoint" (see the *PCS 7 Advanced Process Library* manual). This is used to slow down the setpoint set in the valve position controller.

By default, the gradient limit effect is disabled. If the speed of the valve stroke has to be reduced by the Partial Stroke Test application by limiting the gradient, SP_DnRaLim must be set to a value greater than 0 and SP_RateOn = TRUE. Note that the speed is however only limited when a Partial Stroke Test is performed. If the valve is actuated by a logic other than the Partial Stroke Test logic (e.g. in the case of forced closing of the valve by valve trigger), speed limiting is disabled. When

using the ramp function, the TT and MonTi parameters should be adjusted so that the Partial Stroke Test is performed correctly.

**Note**

For technological reasons, it is not possible to change the parameters of "gradient limit" in the faceplate. They are only shown with their current values. A change is generally made during commissioning by direct entry in the CFC and therefore no change is normally made.

### Opening additional faceplates

You can open the standard view of the technological block (e.g. VlvAnL) interconnected to the input parameter `MV` using a button in the standard view of the faceplate. The interconnection makes the button in the standard view visible.

### SIMATIC BATCH functionality

This block has the standard function "SIMATIC BATCH functionality" (see the *"PCS 7 Advanced Process Library"* manual).

## 4.5.11.4 PST error handling

The block overwrites the following parameters with 0 if they were configured with a negative value and set the output parameter `ParaErr`:

- `TT`
- `TI`
- `MonVal`
- `MonTi`
- `TestHyst`
- `MonVal_S`
- `MonTi_S`
- `DT_S`
- `TT_S`

### Overview of error numbers

The `ErrorNum` I/O can be used to output various error numbers:

| Error ID | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 43 | TimeFactor < 0 or >2 |

### 4.5.11.5 PST reporting

#### Message behavior

The PST block contains the logic and the setup for the messages that can be displayed in the PCS 7 OS. To generate messages, `PST.MsgLock` (lock messages) must be set to 1. Thereafter, the standard settings are displayed for Partial Stroke Test application messages.

| Message instance | Message ID | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 1 | Status message - AS | Partial stroke test started |
| | SIG 2 | Status message - AS | Partial stroke test ended: response time = @4R%4.1f@ sec |
| | SIG 3 | Status message - AS | Partial stroke test ended: test setpoint = @6R%4.1f@%, final position = @5R%4.1f@% |
| | SIG 4 | Status message - AS | Partial stroke test successful |
| | SIG 5 | AS process control message - fault | Partial stroke test failed |
| | SIG 6 | AS process control message - fault | Partial stroke test failed because valve is stuck |
| | SIG 7 | Status message - AS | Partial stroke test aborted |
| | SIG 8 | AS process control message - fault | Partial stroke test feedback error |

| Message instance | Message ID | Message class | Event |
|---|---|---|---|
| MsgEvId2 | SIG 1 | AS process control message - fault | Partial stroke test NOT run: Trip state not met |
| | SIG 2 | AS process control message - fault | Partial stroke test NOT run: Permissives not met |
| | SIG 3 | AS process control message - fault | Partial stroke test NOT run: Feedback bad quality |
| | SIG 4 | AS process control message - fault | Partial stroke test NOT run: Feedback below tolerance |
| | SIG 5 | AS process control message - fault | Partial stroke test NOT run: Solenoid test is running |
| | SIG 6 | AS process control message - fault | Partial stroke test NOT run: Invalid parameter |
| | SIG 7 | Status message - AS | Full stroke test passed |
| | SIG 8 | AS process control message - fault | Full stroke test required |

| Message instance | Message ID | Message class | Event |
|---|---|---|---|
| MsgEvId3 | SIG 1 | AS process control message - fault | Partial stroke test required |
| | SIG 2 | AS process control message - fault | Parameterization error calculation |
| | SIG 3 | AS process control message - fault | Parameter change calculation |
| | SIG 4 | Status message - AS | Solenoid valve test started |
| | SIG 5 | Status message - AS | Solenoid valve test ended |
| | SIG 6 | Status message - AS | Solenoid valve test passed |
| | SIG 7 | AS process control message - fault | Solenoid valve test failed |
| | SIG 8 | AS process control message - fault | Solenoid valve test NOT run |

### Associated values for the message instance `MsgEvId1`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | `RespTime.Value` |
| 5 | `FinalVal.Value` |
| 6 | `TestSP_Out` |
| 7 | Not used |
| 8 | Not used |
| 9 | Not used |
| 10 | Not used |

### Associated values for message instance `MsgEvId2`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | Not used |
| 5 | Not used |
| 6 | Not used |
| 7 | Not used |
| 8 | Not used |
| 9 | Not used |
| 10 | Not used |

### Associated values for message instance `MsgEvId3`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | Not used |
| 5 | Not used |
| 6 | Not used |
| 7 | Not used |
| 8 | Not used |
| 9 | Not used |
| 10 | Not used |

## 4.5.11.6 PST I/Os

### I/Os of PST I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `Aborted_MsgEn` | 1 = Enable "Partial Stroke Test aborted" message | BOOL | 1 |
| `AbortLi` | Abort test<br>(1=Abort)<br>`AbortLi` can be connected to a user-configured logic to abort a Partial Stroke Test. During an active test (i.e. when `F_PST.QRUN` = 1), the test is aborted if `AbortLi` switches to 1. Note that a test is aborted if `AbortLi` switches from 0 to 1 (i.e. `AbortLi` must be reset between two tests). | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `AbortOp` | 1 = Abort Partial Stroke Test via OS operator input | BOOL | 0 |
| `BatchEn` | 1 = Enable allocation | BOOL | 0 |
| `BatchID` | Batch ID | DWORD | 16#00000000 |
| `BatchName` | Batch name | STRING[32] | '' |
| `BlkTypeInfo*` | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| `ChangeCalc` | 1=Change of `PST_Calc` parameter | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `ChangeCalc_MsgEn` | 1 = Enable "Calculation parameter assignment error" message | BOOL | 1 |
| `ClosePos` | Valve position for "completely closed"<br>The setpoint of the valve position controller (`SP`) is set to `ClosePos` if:<br>• the valve trigger status is "active" (i.e. `F_PST.TRIP = 0`). | REAL | 0.0 |
| `DT_S` | Delay time for solenoid valve test (in sec.)<br>When `DT_S` > 0.0, automatic tests are enabled.<br>After a passed Partial Stroke Test has been registered, the delay time starts. After the delay time has completed, the solenoid valve test starts. | REAL | 40.0 |
| `Ended_MsgEn` | 1 = Enable "Partial Stroke Test completed" message | BOOL | 1 |
| `Ended_S_MsgEn` | 1 = Enable "Solenoid valve test completed" message | BOOL | 1 |
| `ErrCalc` | 1 = `PST_Calc` error | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `ErrCalc_MsgEn` | 1 = Enable "Calculation parameter change" message | BOOL | 1 |
| `f_CSF` | Quality of feedback (1=good, 0=bad) | STRUCT: UDT_F_B<br>• DATA: BOOL<br>• PAR_ID: WORD<br>• COMPLEM: WORD | -<br>• 0<br>• 16#0000<br>• 16#FFFF |
| `f_E_TI` | Elapsed interval time [ms] | STRUCT: UDT_F_T<br>• DATA: TIME<br>• PAR_ID: WORD<br>• COMPLEM: DWORD | -<br>• 0<br>• 16#0000<br>• 16#FFFFFFFF |
| `f_E_TI_Hours` | Elapsed interval time [h] | STRUCT: UDT_F_I<br>• DATA: INT<br>• PAR_ID: WORD<br>• COMPLEM: WORD | -<br>• 0<br>• 16#0000<br>• 16#FFFF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| f_E_TT | Elapsed test time [ms] | STRUCT: UDT_F_T<br>• DATA: TIME<br>• PAR_ID: WORD<br>• COMPLEM: DWORD | -<br>• 0<br>• 16#0000<br>• 16#FFFFFFFF |
| f_Enable_S | 1 = Solenoid valve test enabled | STRUCT: UDT_F_B<br>• DATA: BOOL<br>• PAR_ID: WORD<br>• COMPLEM: WORD | -<br>• 0<br>• 16#0000<br>• 16#FFFF |
| f_Failed | 1=Last test failed | STRUCT: UDT_F_B<br>• DATA: BOOL<br>• PAR_ID: WORD<br>• COMPLEM: WORD | -<br>• 0<br>• 16#0000<br>• 16#FFFF |
| f_Failed_S | 1=Last solenoid valve test failed | STRUCT: UDT_F_B<br>• DATA: BOOL<br>• PAR_ID: WORD<br>• COMPLEM: WORD | -<br>• 0<br>• 16#0000<br>• 16#FFFF |
| f_FbkErr | 1=Feedback error | STRUCT: UDT_F_B<br>• DATA: BOOL<br>• PAR_ID: WORD<br>• COMPLEM: WORD | -<br>• 0<br>• 16#0000<br>• 16#FFFF |
| f_FbkIn | Feedback signal from the valve position controller | STRUCT: UDT_F_R<br>• DATA: REAL<br>• PAR_ID: WORD<br>• COMPLEM: DWORD | -<br>• 0.0<br>• 16#0000<br>• 16#FFFFFFFF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| f_Inhibit | 1=Start was inhibited | STRUCT: UDT_F_B<br>• DATA: BOOL<br>• PAR_ID: WORD<br>• COMPLEM: WORD | -<br>• 0<br>• 16#0000<br>• 16#FFFF |
| f_Inhibit_S | 1=Solenoid valve test inhibited | STRUCT: UDT_F_B<br>• DATA: BOOL<br>• PAR_ID: WORD<br>• COMPLEM: WORD | -<br>• 0<br>• 16#0000<br>• 16#FFFF |
| f_Interlock | Test enable signals (1=Enable test, 0=Inhibit test) | STRUCT: UDT_F_B<br>• DATA: BOOL<br>• PAR_ID: WORD<br>• COMPLEM: WORD | -<br>• 0<br>• 16#0000<br>• 16#FFFF |
| f_Passed | 1=Last test passed | STRUCT: UDT_F_B<br>• DATA: BOOL<br>• PAR_ID: WORD<br>• COMPLEM: WORD | -<br>• 0<br>• 16#0000<br>• 16#FFFF |
| f_Passed_S | 1=Last solenoid valve test passed | STRUCT: UDT_F_B<br>• DATA: BOOL<br>• PAR_ID: WORD<br>• COMPLEM: WORD | -<br>• 0<br>• 16#0000<br>• 16#FFFF |
| f_PST_Request | 1=Test required | STRUCT: UDT_F_B<br>• DATA: BOOL<br>• PAR_ID: WORD<br>• COMPLEM: WORD | -<br>• 0<br>• 16#0000<br>• 16#FFFF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| f_Run | Execution status (1=executed, 0=not executed) | STRUCT: UDT_F_B<br>• DATA: BOOL<br>• PAR_ID: WORD<br>• COMPLEM: WORD | -<br>• 0<br>• 16#0000<br>• 16#FFFF |
| f_Run_S | Solenoid valve test execution status (1=executed, 0=not executed) | STRUCT: UDT_F_B<br>• DATA: BOOL<br>• PAR_ID: WORD<br>• COMPLEM: WORD | -<br>• 0<br>• 16#0000<br>• 16#FFFF |
| f_SP_Ok | 1=Setpoint reached | STRUCT: UDT_F_B<br>• DATA: BOOL<br>• PAR_ID: WORD<br>• COMPLEM: WORD | -<br>• 0<br>• 16#0000<br>• 16#FFFF |
| f_Trip | Valve state after trip (1=open, 0=closed) | STRUCT: UDT_F_B<br>• DATA: BOOL<br>• PAR_ID: WORD<br>• COMPLEM: WORD | -<br>• 0<br>• 16#0000<br>• 16#FFFF |
| Failed_MsgEn | 1 = Enable "Partial Stroke Test failed" message | BOOL | 1 |
| Failed_S_MsgEn | 1 = Enable "Solenoid valve test failed" message | BOOL | 1 |
| FbkErr_MsgEn | 1 = Enable "Partial Stroke Test feedback error" message | BOOL | 1 |
| FbkIn_OpScale | OS display area for feedback | STRUCT<br>• High: REAL<br>• Low: Real | -<br>• 100.0<br>• 0.0 |
| Feature | I/O for additional functions | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| FST_Passed | 1=New passed Full Stroke Test | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FST_Passed_MsgEn | 1 = Enable "Full Stroke Test passed" message | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| FST_Request | 1=Full Stroke Test required | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FST_Request_MsgEn | 1=Enable "Full Stroke Test required" message | BOOL | 1 |
| Inhibit_MsgEn | 1=Enable "Partial Stroke Test inhibited" message | BOOL | 1 |
| Inhibit_S_MsgEn | 1=Enable "Solenoid valve test inhibited" message | BOOL | 1 |
| MonTi | Monitoring time for the start of the Partial Stroke Test [s]<br>MonTi is the time interval in which the feedback of the valve must change to a defined value (MonVal). If the valve feedback does not change, the Partial Stroke Test is stopped and registered as failed. | REAL | 1.0 |
| MonTi_S | Monitoring time for solenoid valve test [s]<br>MonTi_S is the time period within which a change to the feedback must be detected during the solenoid valve test.<br>If the feedback value changes within the time period specified by MonTi_S by the value specified by MonVal_S, the solenoid valve test is treated as passed. | REAL | 1.0 |
| MonVal | Change in feedback required for start of the Partial Stroke Test<br>MonVal is the value by which the valve feedback must change in a defined time interval (MonTi).<br>If the valve feedback does not change, the Partial Stroke Test is stopped and registered as failed. | REAL | 1.0 |
| MonVal_S | Required change of feedback for solenoid valve test<br>MonVal_S is the value by which the feedback must change during the solenoid valve test.<br>If the feedback value changes within the time period specified by MonTi_S by the value specified by MonVal_S, the solenoid valve test is treated as passed. | REAL | 1.0 |
| MsgEvID1 | Message block message ID 1 | DWORD | 16#00000000 |
| MsgEvID2 | Message block message ID 2 | DWORD | 16#00000000 |
| MsgEvID3 | Message block message ID 3 | DWORD | 16#00000000 |
| MsgLock | 1=Lock messages<br>The PST block contains messages for the Partial Stroke Test application that are used with a PCS 7 OS. Value 1 for MsgLock enables message creation; value 0 for MsgLock inhibits message creation. | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NextFST | Time of next FST | STRING[16] | '' |
| Occupied | 1 = In use by a batch | BOOL | 0 |
| OnOp | 1 = "On" mode via operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp | 1 = "Out of service", via OS operator input | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OpenPos | Valve position for "completely open"<br>• The setpoint (SP) of the valve position controller is set to OpenPos if:<br>the valve trigger status is "not active" (i.e. F_PST.TRIP = 1) and no Partial Stroke Test is in progress<br>• The interlock is lost while a Partial Stroke Test is in progress (i.e. F_PST.ILOCK = 0) | REAL | 100.0 |
| OS_Perm | I/O for operator control permissions | STRUCT<br>• Bit 0: BOOL<br>• ..<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| Passed_MsgEn | 1 = Enable "Partial Stroke Test passed" message | BOOL | 1 |
| Passed_S_MsgEn | 1 = Enable "Solenoid valve test passed" message | BOOL | 1 |
| PST_Request_MsgEn | 1 = Enable "Partial Stroke Test required" message | BOOL | 1 |
| RunUpCyc | Time delay: Number of ramp up cycles | INT | 10 |
| SampleTime | Sample acquisition time [s] | REAL | 0.1 |
| SelFp1 | 1 = Call of a block stored in this parameter as an additional face-plate in the standard view | ANY | " |
| SelFp2 | 1 = Call of a block stored in this parameter as an additional face-plate in the standard view | ANY | " |
| SP_DnRaLim | Limit (bottom) for the setpoint gradient [%/s] | REAL | 100.0 |
| SP_OpScale | OS display area for setpoint | STRUCT<br>• High: REAL<br>• Low: Real | -<br>• 100.0<br>• 0.0 |
| SP_RateOn | 1 = Enable limit of the setpoint gradients | BOOL | 0 |
| SP_UpRaLim | Gradient limit (high) for the setpoint [%/s] | REAL | 100.0 |
| ST_WorstIn | Worst signal status of PST_OrQC | BYTE | 16#80 |
| Started_MsgEn | 1 = Enable "Partial Stroke Test started" message | BOOL | 1 |
| Started_S_MsgEn | 1 = Enable "Solenoid valve test started" message | BOOL | 1 |
| StartLi | Connectable input for manual test request<br>(1=Start test)<br>StartLi can be connected to a user-configured logic to start a Partial Stroke Test. If all test preconditions are met, a Partial Stroke Test is started when StartLi is set to 1. Note that a new test is started if StartLi switches from 0 to 1 (i.e. StartLi must be reset between two tests). | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StartOp | 1 = Start Partial Stroke Test via OS operator input | BOOL | 0 |
| StepNo | Batch step number | DWORD | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| TestHyst | Hysteresis for "Test passed/failed"<br>The success/failure status of a Partial Stroke Test is determined by TestHys. For a passed test, the feedback of the valve position controller must be less than or equal to the value for (TestSP + TestHyst). | REAL | 0.0 |
| TestSP | Valve stroke position<br>TestSP is the setpoint for the valve position that is used for a Partial Stroke Test. During configuration, TestSP must be set to the value that corresponds to the supplier's recommendations for a passed Partial Stroke Test.<br>The value for TestSP should be between the values for the fully open position (OpenPos) and the fully closed position (ClosePos).<br>The Partial Stroke Test logic does not contain position limitation (which prevents the valve closing too far). The user should take this into account when specifying the value for TestSP. | REAL | 100.0 |
| TestSP_HiLim | Limit (high) of the setpoint | REAL | 100.0 |
| TestSP_LoLim | Limit (low) of the setpoint | REAL | 0.0 |
| TestSP_Range | Range for the test setpoint<br>TestSP_Range is the range in which the test setpoint TestSP fluctuates.<br>The setpoint range TestSP_Range must be less than (OpenPos - ClosePos) / 2.0. | REAL | 0.0 |
| TI | Test interval [h]<br>When TI > 0, automatic tests are enabled.<br>If TI is changed while a timer is running, the timer is fully reset (at the same time, no test is run until TI has fully completed).<br>If TI is changed while a timer is running, the user is recommended to perform a manual Partial Stroke Test to take into account the time interval before TI changed.<br>If a recommended Partial Stroke Test is not performed because it has been inhibited (because a specific start precondition has not been met), the timer for the automatic test is re-started and must complete before it is possible to attempt to start a test automatically again. | INT | 0 |
| TimeFactor | Time unit:<br>0 = Seconds<br>1 = Minutes<br>2 = Hours | INT | 0 |
| TT | Duration of the test; maximum valve response time [s]<br>TT is the period of time that is allocated to the valve position setpoint TestSP for a Partial Stroke Test. To determine the TT value, the valve response time should be determined first via a manual stroke test (i.e. the time in which the valve normally reaches TestSP).<br>When TT = 0, a test is not possible. | REAL | 10.0 |
| TT_S | Duration of the solenoid valve test [cycles]<br>TT_S is the time period measured in cycles for which the solenoid valve receives closing pulses during a solenoid valve test.<br>For a solenoid valve test to be run, it is necessary for TT_S > 0. | INT | 3 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#0 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Abort | Abort test (1=Abort) | BOOL | 0 |
| AbortedAct | 1=Test was aborted | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Active | 1=Test active | BOOL | 0 |
| ClosePosOut | Valve position for "completely closed" | REAL | 0.0 |
| DT_S_Out | Delay time for solenoid valve test [ms] | TIME | 0ms |
| E_Time | Elapsed test time [s] | STRUCT<br>• VALUE: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ErrorNum | Output of the existing error number. For error numbers that can be output by this block, see AUTOHOTSPOT. | INT | -1 |
| Failed_S_Act | 1=Last solenoid valve test failed | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FailedAct | 1=Last test failed | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkErrAct | 1=Feedback error | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkOut | Feedback signal from the valve position controller | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FinalVal | Valve actual position at end of test | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FST_PassedAct | 1=New passed Full Stroke Test | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| GrpErr | Quality of feedback (1=good, 0=bad) | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Inhibit_S_Act | 1=Solenoid valve test inhibited | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| InhibitAct | 1=Start was inhibited | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| LockAct | Test enable signals (1=Enable test, 0=Inhibit test) | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MonTi_S_Out | Monitoring time for solenoid valve test [ms] | TIME | 0ms |
| MonTiOut | Monitoring time for start Partial Stroke Test [ms] | TIME | 0ms |
| MonVal_S_Out | Change of feedback value for solenoid valve test required | REAL | 0.0 |
| MonValOut | Change in feedback value within the monitoring time required | REAL | 0. |
| MsgAckn1 | Message block 1: Output value for ACK_STATE | WORD | 16#0000 |
| MsgAckn2 | Message block 2: Output value for ACK_STATE | WORD | 16#000 |
| MsgAckn3 | Message block 3: Output value for ACK_STATE | WORD | 16#0000 |
| MsgErr1 | Message error 1 (output ERROR of first ALARM_8P) | BOOL | 0 |
| MsgErr2 | Message error 1 (output ERROR of first ALARM_8P) | BOOL | 0 |
| MsgErr3 | Message error 1 (output ERROR of first ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message block 1: Output value for STATUS | WORD | 16#0000 |
| MsgStat2 | Message block 2: Output value for STATUS | WORD | 16#0000 |
| MsgStat3 | Message block 3: Output value for STATUS | WORD | 16#0000 |
| NextPST | Time stamp for next Partial Stroke Test<br>NextPST is only used for display to the user. The logic to calculate the next required automatic Partial Stroke Test is executed by the F_PST block. | STRING[16] | '' |
| OnAct | 1 = Block is "On" | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpenPosOut | Valve position for "completely open" | REAL | 0.0 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS1PermLog | Display of OS1Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1Perm | DWORD | 16#FFFFFFFF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ParaErr | Indicates invalid parameters<br>(1=Invalid parameters)<br>A Partial Stroke Test cannot be started if one of the test parameters (for times, values or ranges) has an invalid value. The following validity checks are run for a Partial Stroke Test:<br>• OpenPos > ClosePos<br>• TT > 0<br>• MonVal >= 0 and MonVal < (OpenPos – TestSP)<br>• MonTi >= 0 and MonTi < TT<br>• TestHyst >= 0<br>• (TestSP + TestSP_Range + TestHyst) < OpenPos and (TestSP – TestSP_Range – TestHyst) > ClosePos<br>A solenoid valve test cannot be started if one of the test parameters (for times, values or ranges) has an invalid value. The following validity checks are run for a solenoid valve test:<br>• DT_S >= 0 and DT_S < (TI – TT_S – MonTi_S)<br>• TT_S > 0 and TT_S < (TI – MonTi_S)<br>• MonVal_S >= 0 and MonVal_S < OpenPos<br>• MonTi_S >= 0 and MonTi_S < TT_S<br>If an invalid parameter is detected, ParaErr is set t o1. | BOOL | 0 |
| Passed_S_Act | 1=Last solenoid valve test passed | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PassedAct | 1=Last test passed | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PST_ReqAct | 1=Partial Stroke Test required | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RespTime | Response time until test setpoint reached [s] | STRUCT<br>• VALUE: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Run_S_Act | Solenoid valve test execution status (1=executed, 0=not executed) | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RunAct | Execution status (1=executed, 0=not executed) | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP | Setpoint of the valve position controller | STRUCT<br>• VALUE: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP_DnRaAct | 1 = Negative gradient limit of the setpoint is active | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_RateTarget | Target setpoint for gradient limit | REAL | 0.0 |
| SP_UpRaAct | 1 = Positive gradient limit of the setpoint is active | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Start | Manual test requirement (1=start test) | BOOL | |
| StartEn_S_Act | 1 = Solenoid valve test enabled | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StartEnAct | Status: 1 = Partial Stroke Test enabled | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Status1 | Status word | DWORD | 16#0 |
| Status2 | Status word | DWORD | 16#0 |
| TestHystOut | Valve position hysteresis for "Test passed/failed" | REAL | 0.0 |
| TestSP_HiOut | Limit (high) of the setpoint | STRUCT<br>• VALUE: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| TestSP_LoOut | Limit (low) of the setpoint | STRUCT<br>• VALUE: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| TestSP_Out | Valve stroke position | REAL | 0.0 |
| TestTS | Time stamp for last test | STRING[16] | '' |
| TI_CalcOut | Test interval for PST_Calc [h] (0=no automatic tests) | INT | |
| TI_Out | Test interval [h] (0=no automatic tests) | INT | 0 |
| TripAct | Valve state after trip (1=open, 0=closed) | STRUCT<br>• VALUE: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| TT_Out | Duration of the test; maximum valve response time [ms] | TIME | 0ms |
| TT_S_Out | Duration of the solenoid valve test [cycles] | INT | 0 |

### 4.5.11.7 Operator control and monitoring

#### Operator control and monitoring of the Partial Stroke Test

At runtime, the user can access the Partial Stroke Test component user interface via the PCS 7 OS. The block icon and the faceplate for the Partial Stroke Test application in this case are represented visually in the application configured and monitored in SIMATIC Manager.

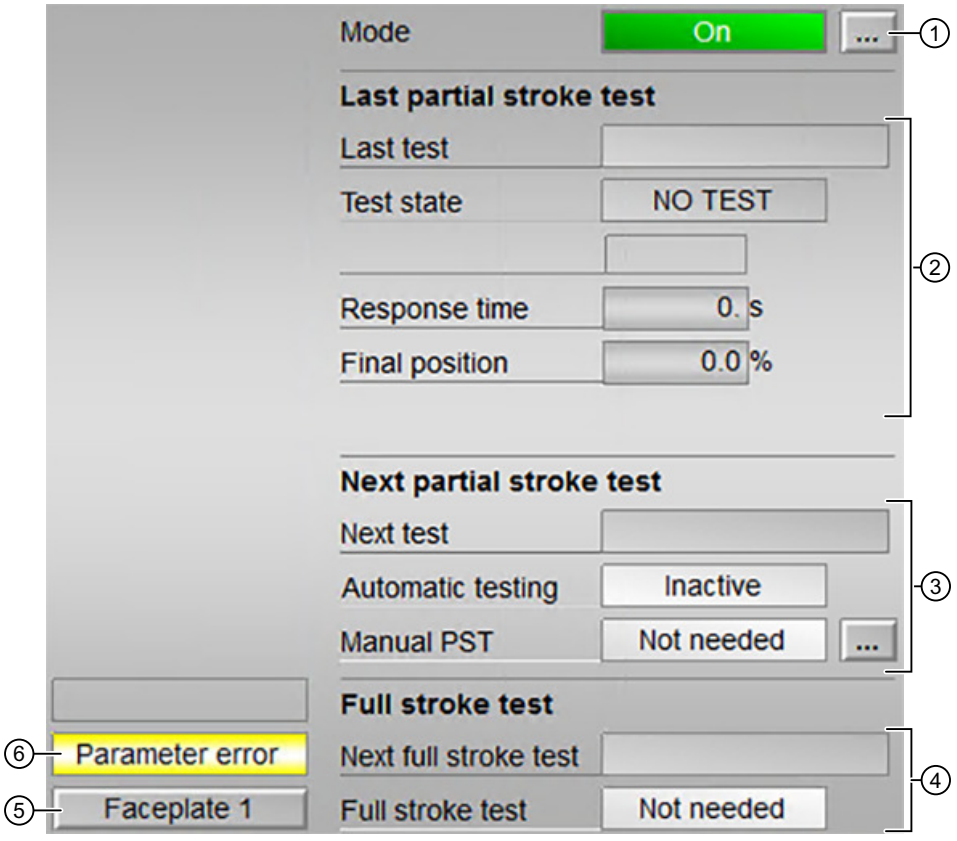
#### Views of the Partial Stroke Test

The PST block has the following views:

- Standard view of PST (Page 687)
- Alarm View (Page 336)
- Parameter view (Page 689)
- Ramp view (Page 690)
- Preview (Page 691)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for PumpMon (Page 692)

For general information on the faceplate and block icon, refer to the "Structure of the faceplate (Page 279)" and "Block icon structure (Page 260)" sections.

### Partial Stroke Test standard view




**(1) Display and switchover of operating mode**
This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On
- Out of service

**(2) Last Partial Stroke Test**
This area display the result of the last performed Partial Stroke Test.

The following information is available:

- "Last test": Time of last Partial Stroke Test (`TestTS`)
- "Test status": Result of the last Partial Stroke Test
  - "NO TEST": No Partial Stroke Test has yet been performed
  - "RUNNING": A Partial Stroke Test is running now
  - "PASSED": The last Partial Stroke Test passed
  - "FAILED": The last Partial Stroke Test failed
  - "INHIBITED": The last Partial Stroke Test was inhibited
  - "ABORTED": The last Partial Stroke Test was aborted
  - "FEEDBACK ERROR": The last Partial Stroke Test had a feedback error

- "Test time": Current time period of a running Partial Stroke Test (`E_Time`). This field is only visible if "Test status" = "RUNNING".
- "Response time": Response time to reach the test setpoint during the last Partial Stroke Test (`RespTime`)
- "End position": End position when test setpoint was reached during the last Partial Stroke Test (`FinalVal`)
- "Solenoid valve status": Result of the last solenoid valve test
  - "NO TEST": No solenoid valve test has yet been performed
  - "RUNNING": A solenoid valve test is running now
  - "PASSED": The last solenoid valve test passed
  - "FAILED": The last solenoid valve test failed
  - "INHIBITED": The last solenoid valve test was inhibited

  This field is only visible if the optional visible has been enabled (`StartEn_S_Act`)

**(3) Next Partial Stroke Test**
This area displays information about the next planned Partial Stroke Test.

The following information is available:

- "Next test": Time (`NextPST`) of the next planned Partial Stroke Test (time stamp is only shown if automatic test is enabled)
- "Automatic test":
  - "Active": Calculation of the next automatic test time is enabled (`TI > 0`)
  - "Inactive": No calculation of the next test time (`TI = 0`)
- "Manual PST": Provides the ability to start or inhibit a manual Partial Stroke Test. The request status for a manual test is also displayed:
  - "Not required": A manual Partial Stroke Test is not required
  - "REQUESTED": A manual Partial Stroke Test must be performed (if "Automatic test" = "Active" and `F_PST.AUTO_EN = FALSE`)

**(4) Full Stroke Test**
This area displays information about the next Full Stroke Test that is necessary.

The following information is available:

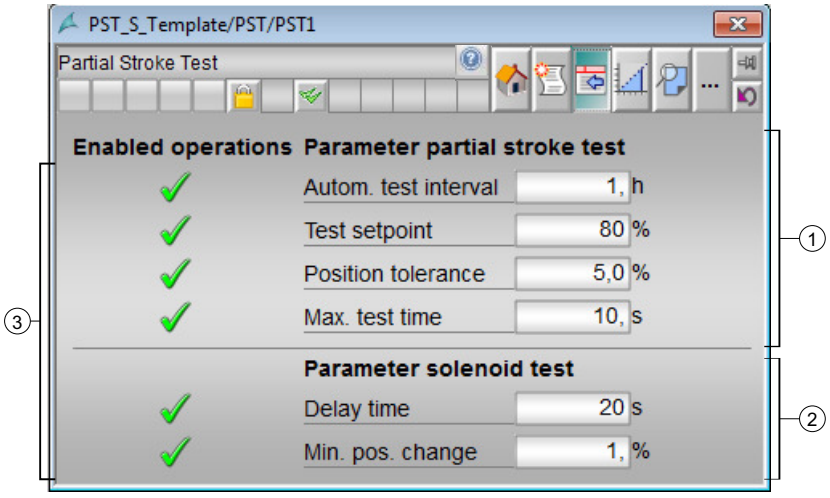
- "Next Full Stroke Test": Indicates the time (`NextFST`) of the next required Full Stroke Test (if "Automatic test" = "Active")
- "Full Stroke Test": Indicates the request status for a Full Stroke Test:
  - "Not required": A Full Stroke Test is currently not required
  - "REQUESTED": A Full Stroke Test must be performed

**(5) Tab key for switching to the standard view of any faceplate**
This display is only visible when the corresponding `SelFp1` block input is connected. Use this jump button to reach the standard view of the interconnected block.

**(6) Display area for block states**
This area provides additional information on the operating state of the block:

- Parameter assignment error
- Internal Error

## Partial Stroke Test parameter view




**(1) Partial Stroke Test parameters**
You can enter the parameters for the Partial Stroke Test in this area.

You can change the following parameters:

- "Automatic test interval": Indicates the test interval for automatic performance of Partial Stroke Tests in [h] (`TI`)
- "Test setpoint": Indicates the setpoint (`TestSP`)
- "Position tolerance": Indicates the position tolerance about the position setpoint (`TestHyst`)
- "Maximum test time": Indicates the maximum time period to reach the test setpoint (`TT`)

**(2) Solenoid valve test parameters**
You can enter the parameters for the optional solenoid valve test here.

You can change the following parameters if the solenoid valve test is enabled (`StartEn_S_Act`):

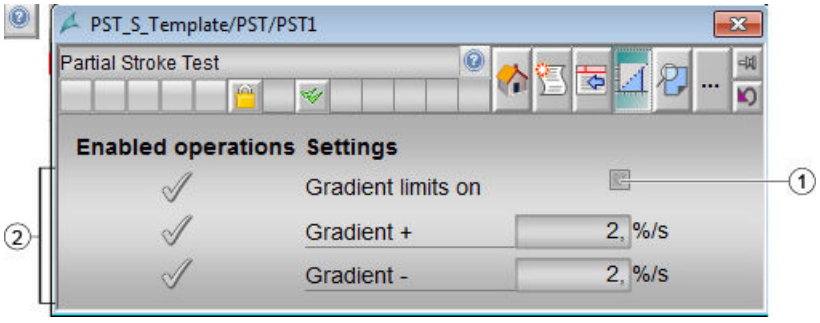
- "Delay time": Indicates the time delay between a passed Partial Stroke Test and the start of a solenoid valve test (`DT_S`)
- "Minimum change in position": Indicates the value of the change in position (`MonVal_S`) that is required to pass the solenoid valve test within the defined test time (`MonTi_S`).

**(3) Enabled operations**
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark:** The OS operator can control this parameter
- **Gray check mark:** Access to this parameter is temporarily disabled for the OS operator due to process conditions
- **Red cross:** The OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

## Partial Stroke Test ramp view




**(1) Enable "Gradient limit"**
The option checkbox shows you whether the "Gradient limit" is enabled and which values the "Gradient limit" has for positive and negative changes to the setpoint.

**Note**

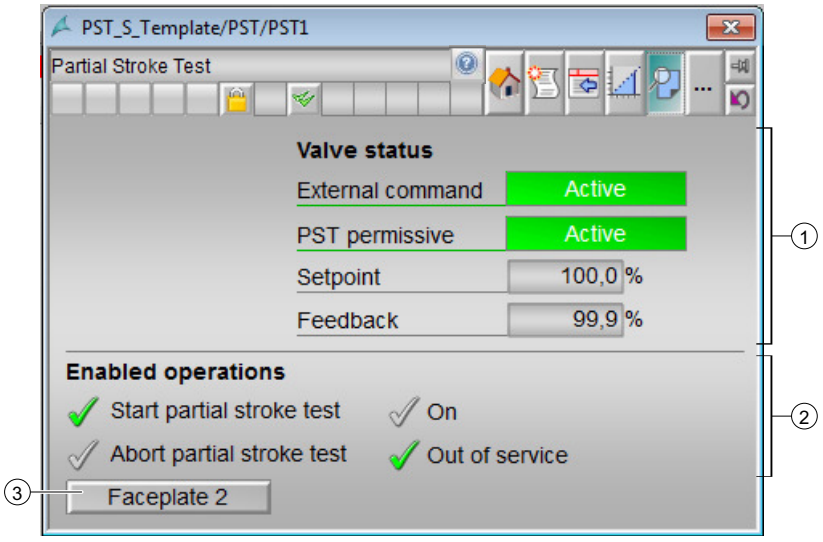
For technological reasons, it is not possible to change the parameters of "gradient limit" in the faceplate. They are only shown with their current values. A change is generally made during commissioning by direct entry in the CFC and therefore no change is normally made.

**(2) Enabled operations**
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark:** The OS operator can control this parameter
- **Gray check mark:** Access to this parameter is temporarily disabled for the OS operator due to process conditions
- **Red cross:** The OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

## Partial Stroke Test preview




**(1) "Valve status" preview area**
A preview is displayed here for the following values:

- "External command": Indicates the valve control status (`TripAct`)
- "Enable status": Indicates the status of the enable conditions for a Partial Stroke Test (`LockAct`)
- "Setpoint": Indicates the currently calculated setpoint (`SP`)
- "Read back value": Indicates the position feedback currently read back (`FbkOut`)

**(2) Enabled operations**
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark:** The OS operator may edit this parameter
- **Gray check mark:** Editing of this parameter is temporarily disabled for the OS operator due to process conditions
- **Red cross:** The OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`) .

The following enabled operations are shown here:

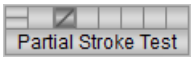
- "Start Partial Stroke Test": You may start the Partial Stroke Test.
- "Abort Partial Stroke Test": You may abort the Partial Stroke Test.
- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.

### Partial Stroke Test block icon

A block icon with the following displays is available to you:

- Block type or process tag
- Operating modes
- Current status of the Partial Stroke Test application.

The block icon from template @TemplateAPLV9.PDL:

| Symbols | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in full display in the normal state |
|  | 2 | Block icon in full display in the normal state |
|  | | Block icon in the "Out of service" operating mode |

You can find additional information about the block icon and the operator control options in the block icon in the following sections:

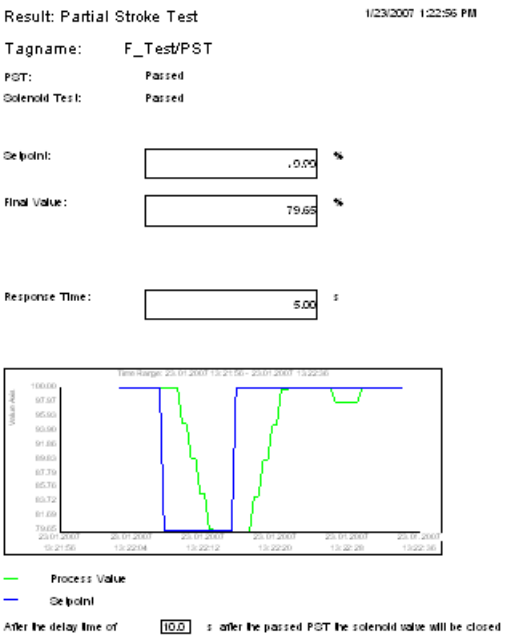
- "Configuring the block icons (Page 267)"
- "Block icon structure (Page 260)"
- "Operation via the block icon (Page 269)".

## 4.5.12 Create reports

The user has the option of creating reports after a Partial Stroke Test has been performed. The following files are provided for this purpose:

- Report layouts
  The report layouts are located in WinCC Explorer under "Report Designer\Layouts":
  - PST_DEU.RPL (German version)
  - PST_ENU.RPL (English version)
- PST.pas action for starting the print job.
  This file is located in the WinCC installation folder in the folder \WinCC\CML\PST after the Condition Monitoring Library has been installed.

Layout of the test document for the Partial Stroke Test:



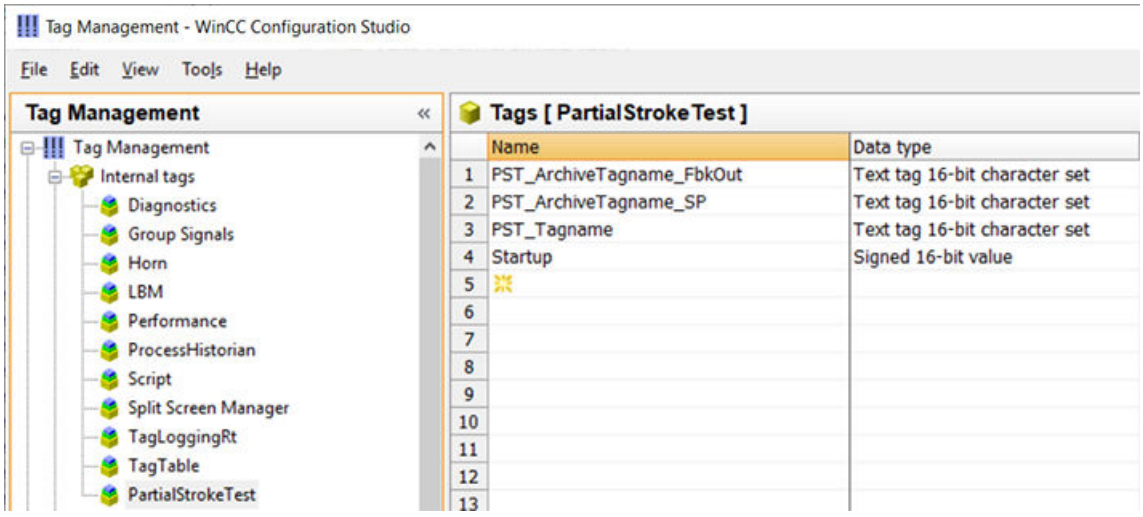

The following sections describe the procedure for configuring and activating a report for a Partial Stroke Test.

## Internal Tags

The internal tags are required to transfer the tag names and names of the archive tags to the report layout.

1. Open the Tag Management.
2. Add a new tag groups named "PartialStrokeTest" to the internal tags.

3. Add four tags with the below names and data types to the tag group PartialStrokeTest.
   - PST_ArchiveTagname_FbkOut - text tag 16-bit character set
   - PST_ArchiveTagname_SP - text tag 16-bit character set
   - PST_Tagname - text tag 16-bit character set
   - Startup - Signed 16-bit value
4. Assign the initial value of 2 to the "Startup" tag.



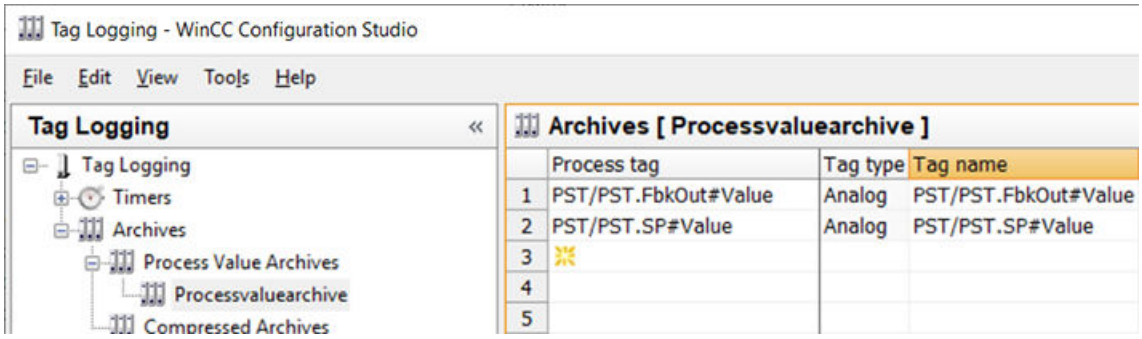

## Process tags

The report layout shows a graphic curve made up of the feedback value and the setpoint for the Partial Stroke Test that has been performed. The curve is based on data from a process data archive.

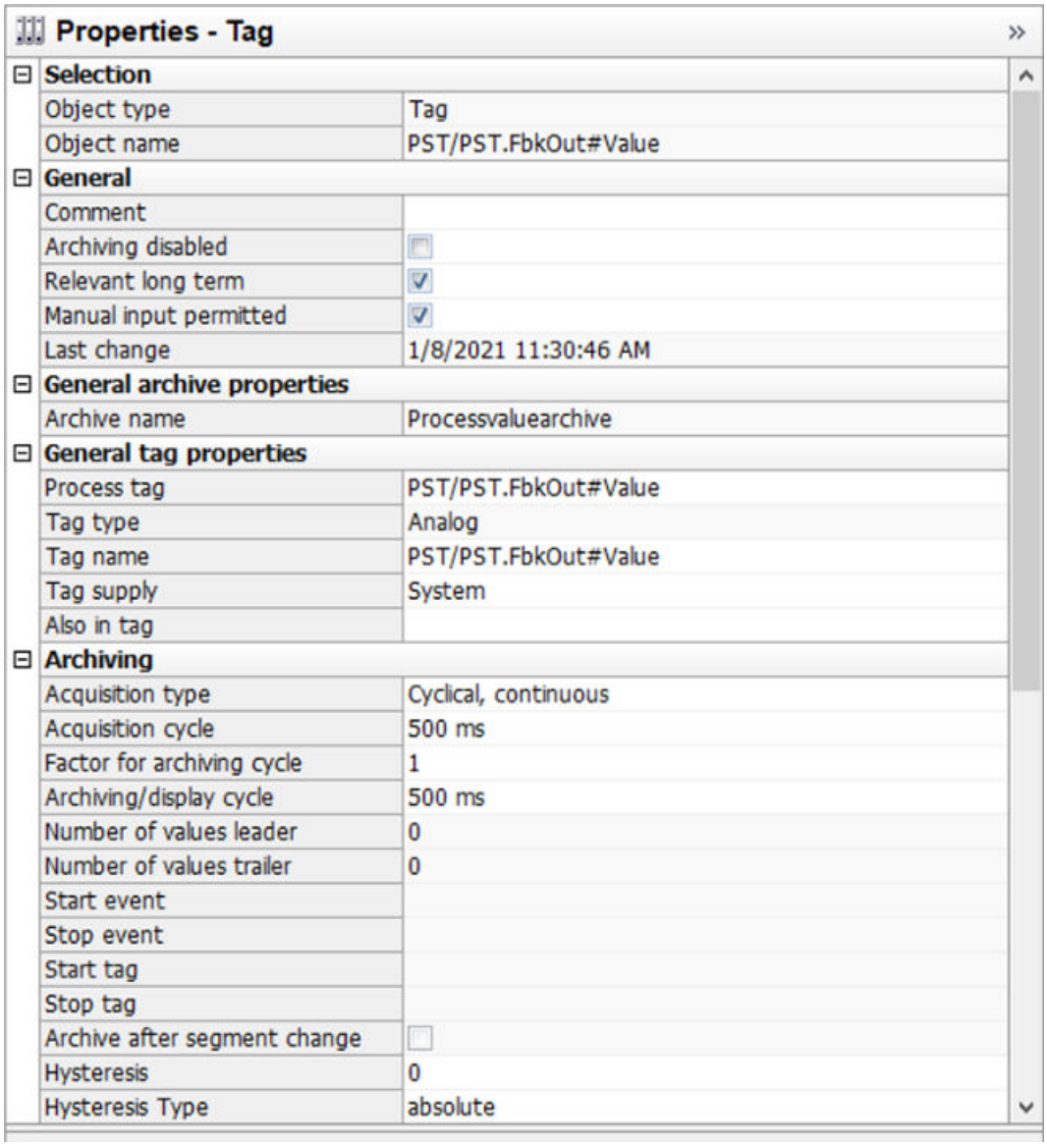
Proceed as follows to create the archive:

1. Open the Tag Logging.
2. Insert a new process value archive named **"Process value archive"**.

3. Add the `FbkOut` and `SP` tags of the PST block to the process data archive. The tag names of these process tags must not be changed!




4. The properties of the process tags must be changed.
   Set the "Archiving type" to "Cyclic, selective" in the "Archiving" tab.
   As the "Start tag", select the `.Active` I/O of the corresponding PST block.



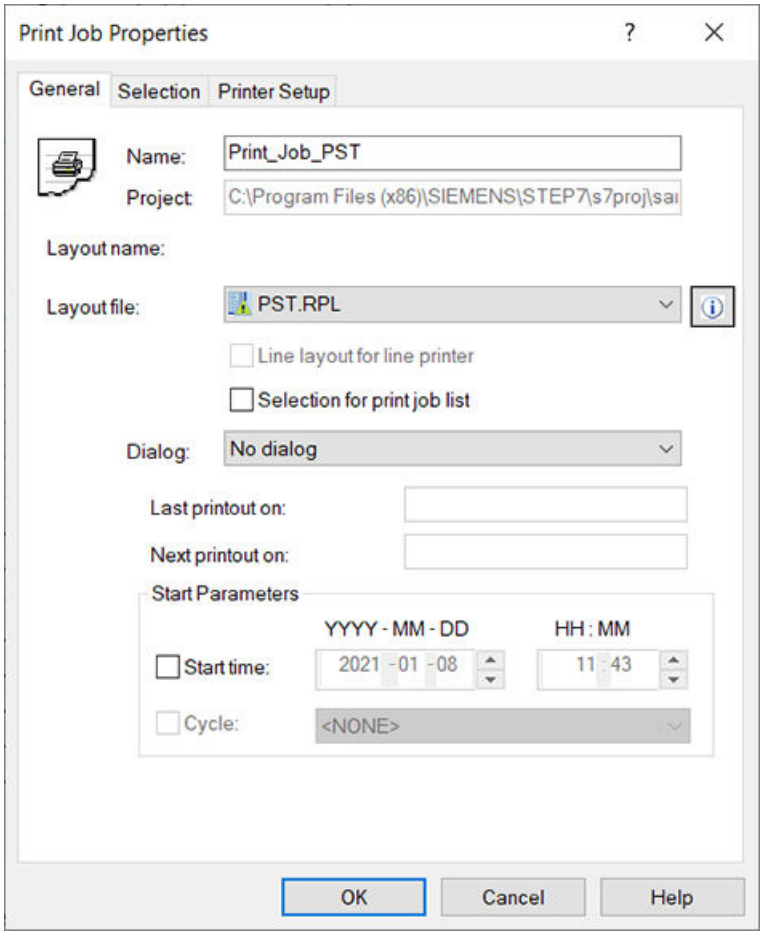

## Creating print jobs

Open WinCC Explorer → "Report Designer" → "Print jobs" and add a new print job.

The print job must be configured as described below:

- Name → Print_Job_PST
- Layout → PST.rpl

For all other language versions, the names of the print jobs and the report layout can be configured freely.




## Activate print job

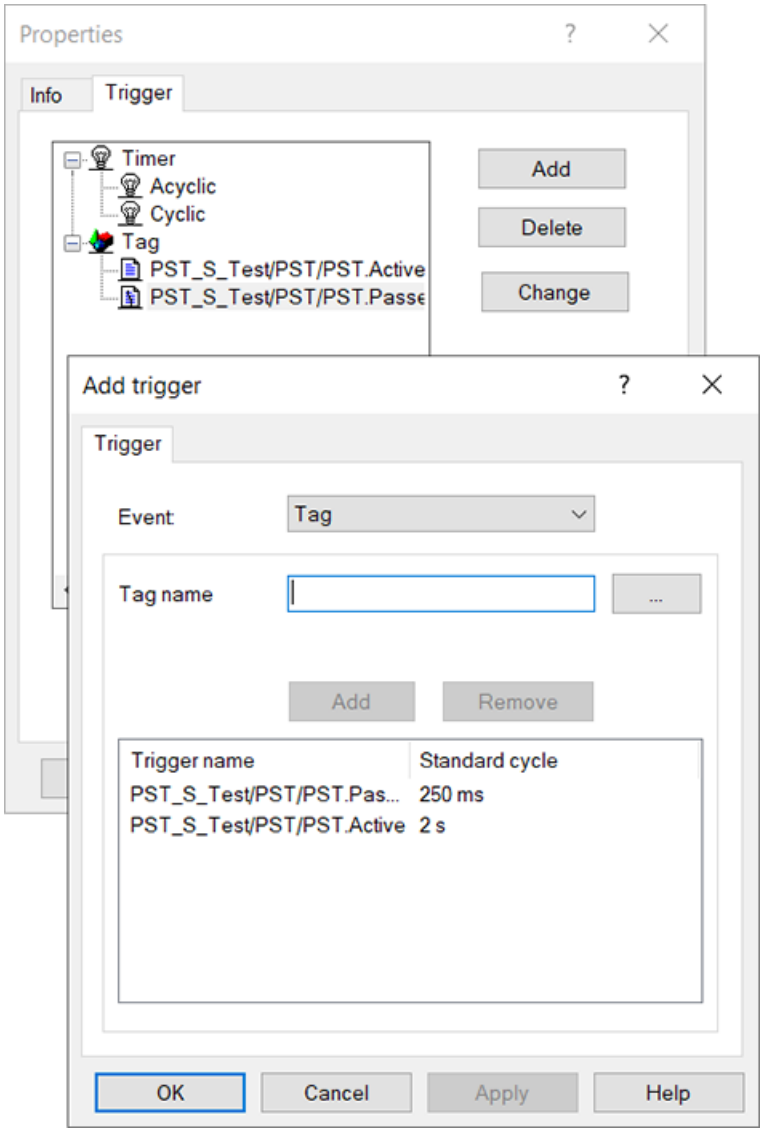
The print job should be started when a Partial Stroke Test and, if applicable, a solenoid valve test have been performed and complete. The global script action PST.pas that is found in the WinCC installation folder \WinCC\CML\PST is used as a template to trigger a print job for a PST instance.

The global script action PST.pas must be copied to the folder <...projectname\wincproj \OS_project_name\computer_name\PAS> and be adapted for the other PST function blocks.

Adaptations to PST.pas:

1. Open the C editor for the local action "PST.pas".
2. Define the trigger tag: Click on this icon or select "Edit" → "Info".

3. Replace the existing tags in the properties windows with the tags of the applicable Partial Stroke Test block.
   The standard cycle for the "`.PassedAct#Value`" tag must be less than or equal to the execution cycle in the CPU.



4. Change the assignment of the Partial Stroke Test tag name in the source code (circled character string) to the tag name of your PST:

```
#include "apdefap.h"

int gscAction( void )
{
BOOL active;
int startup;

char tagname[260];
char archivetagname_PST[260];

char tagname_PST[260] = "PST_S_Test/PST/PST"; // Has to be edited by user: tagname of PST

//Set tagname for report
SetTagChar ("PST_Tagname", tagname_PST);

//Set archive tagname of FbkOut for report
sprintf (archivetagname_PST, "Processvaluearchive\\%s.FbkOut#Value", tagname_PST);
//sprintf (archivetagname_PST, "Prozesswertarchiv\\%s.FbkOut#Value", tagname_PST);
SetTagChar ("PST_ArchiveTagname_FbkOut", archivetagname_PST);

//Set archive tagname of SP for report
sprintf (archivetagname_PST, "Processvaluearchive\\%s.SP#Value", tagname_PST);
//sprintf (archivetagname_PST, "Prozesswertarchiv\\%s.SP#Value", tagname_PST);
SetTagChar ("PST_ArchiveTagname_SP", archivetagname_PST);

sprintf (tagname, "%s.Active", tagname_PST);
active = GetTagBit (tagname);
startup = (int) GetTagWord ("Startup");

if (startup == 0)
        {
        if (!active)
                {
                        RPTJobPreview ("Print_Job_PST");
                        //RPTJobPrint ("Print_Job_PST");
                }
        }
else
{
        SetTagWord ("Startup", (WORD) (startup - 1));
}

return 0;
}
```

5. Adjust the function call for the print function:
   - Open "RPTJobPreview" to start the print preview, which can then be printed
   - Open "RPTJobPrint" to print the print job immediately.

# Monitoring blocks

# 5

## 5.1 Comparison of large & small blocks

### 5.1.1 MonAnL compared to MonAnS

#### Comparison of the MonAnL and MonAnS blocks

The following tables are intended to help you decide which block to use.

#### Memory and runtime savings of the small block compared to the large block

You save the following resources for each instance:

- Memory space: ~ 45%
- Runtime: ~ 30%

#### Operating modes of the blocks

| | MonAnL | MonAnS |
|---|---|---|
| On (Page 77) | X | X |
| Out of service (Page 77) | X | X |

#### Functions of the blocks

| | MonAnL | MonAnS |
|---|---|---|
| Alarm delays with two time values per limit pair (Page 227) | X | |
| Alarm delays with one time value per limit pair (Page 226) | | X |
| Limit monitoring (Page 93) | X | X |
| Suppressing messages using the MsgLock parameter (Page 231) | X | X |
| Gradient monitoring (Page 719) | X | |
| Displaying auxiliary values (Page 239) | X | |
| Forming and outputting the signal status for technologic blocks (Page 117) | X | X |

| | MonAnL | MonAnS |
|---|---|---|
| Dead band (Page 67) | X | X |
| Release for maintenance (Page 70) | X | X |
| Simulating signals (Page 64) | X | X |
| Selecting a unit of measure (Page 240) | X | X |
| Operator control permissions (Page 285) | X | X |
| Generating instance-specific messages (Page 230) | X | X |
| Display and operator input area for process values and setpoints (Page 233) | X | X |
| Opening additional faceplates (Page 234) | X | X |
| Time stamp (Page 2046) | X | |
| SIMATIC BATCH functionality (Page 73) | X | X |
| Providing PV limit at the output (Page 719) | x | |

## Configurable functions using the `Feature` parameter

| Bit number | `Feature` bit function | MonAnL | MonAnS |
|---|---|---|---|
| 0 | Set startup characteristics (Page 148) | X | X |
| 1 | Reaction to the out of service mode (Page 188) | X | X |
| 22 | Update acknowledgment and error status of the message call (Page 171) | X | |
| 24 | Enabling local operator authorization (Page 169) | X | X |
| 25 | Suppression of all messages (Page 185) | X | X |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) | X | X |
| 28 | Disabling operating points (Page 156) | X | |
| 29 | Signaling limit violation (Page 181) | X | |

## 5.1.2 MonDiL compared to MonDiS

### Comparison of the MonDiL and MonDiS blocks

The following tables are intended to help you decide which block to use.

### Memory and runtime savings of the small block compared to the large block

You save the following resources for each instance:

- Memory space: ~ 55%
- Runtime: ~ 12%

### Operating modes of the blocks

| | MonDiL | MonDiS |
|---|---|---|
| On (Page 77) | x | x |
| Out of service (Page 77) | x | x |

### Functions of the blocks

| | MonDiL | MonDiS |
|---|---|---|
| Suppression and and reporting of signal flutter (Page 774) | x | |
| Delaying the on and off function (Page 774) | x | |
| Delaying the on function (Page 800) | | x |
| Specifying the status display for the block icon (Page 774) | x | x |
| Displaying auxiliary values (Page 239) | x | |
| Labeling of buttons and text (Page 235) | x | x |
| Generating instance-specific messages (Page 230) | x | x |
| Suppressing messages using the MsgLock parameter (Page 231) | x | x |
| Forming and outputting the signal status for technologic blocks (Page 117) | x | x |
| Release for maintenance (Page 70) | x | x |
| Simulating signals (Page 64) | x | x |

| | MonDiL | MonDiS |
|---|---|---|
| Operator control permissions (Page 285) | x | x |
| Opening additional faceplates (Page 234) | x | x |
| Time stamp (Page 2046) | x | |
| SIMATIC BATCH functionality (Page 73) | x | x |
| Group display for limit monitoring, CSF and ExtMsgx (Page 92) | x | x |

## Configurable functions using the `Feature` parameter

| Bit number | `Feature` bit function | MonDiL | MonDiS |
|---|---|---|---|
| 0 | Set startup characteristics (Page 148) | x | x |
| 1 | Reaction to the out of service mode (Page 188) | x | x |
| 22 | Update acknowledgment and error status of the message call (Page 171) | x | |
| 24 | Enabling local operator authorization (Page 169) | x | x |
| 25 | Suppression of all messages (Page 185) | x | x |

# 5.2 AV - Displaying and monitoring additional value

## 5.2.1 Description of AV

### Object name (type + number) and family

Type + number: FB 1903

Family: Monitor

### Area of application for AV

The block is used for the following applications:

- Monitoring an additional analog value at a technologic block (motors or valves with motor).

### How it works

The block must be connected to a channel block and monitors an additional analog value. The messages of the AV block appear in the alarm view of the technologic block connected to it. Monitoring limits are configured and controlled at the technologic block.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100). The AV block and the technologic block must be installed in the same cyclic interrupt OB.

Interconnect the `AV_Tech` output parameter of the AV block to the input parameter `AV` of the technologic block in the CFC.

**Note**

Interconnection of the block to multiple technologic blocks is not permitted.

For the AV block, the Advanced Process Library contains process tag type templates; these serve as examples by providing various application scenarios for this block.

Examples of process tag types:

- Motor with an additional analog value and time-stamped signals (Motor_AV_EventTs) (Page 2863)

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section AV I/Os (Page 711).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `MsgLock` |
| 4 | `AV_AH_Act.Value` |
| 5 | `AV_WH_Act.Value` |
| 6 | `AV_TH_Act.Value` |
| 7 | `AV_TL_Act.Value` |
| 8 | `AV_WL_Act.Value` |
| 9 | `AV_AL_Act.Value` |
| 10 | `AV_AH_En` |
| 11 | `AV_WH_En` |
| 12 | `AV_TH_En` |
| 13 | `AV_TL_En` |
| 14 | `AV_WL_En` |
| 15 | `AV_AL_En` |
| 16 | `AV_AH_MsgEn` |
| 17 | `AV_WH_MsgEn` |
| 18 | `AV_TH_MsgEn` |
| 19 | `AV_TL_MsgEn` |
| 20 | `AV_WL_MsgEn` |
| 21 | `AV_AL_MsgEn` |
| 22 | Delay of `AV_AH_Act` |
| 23 | Delay of `AV_WH_Act` |
| 24 | Delay of `AV_TH_Act` |
| 25 | Delay of `AV_TL_Act` |
| 26 | Delay of `AV_WL_Act` |
| 27 | Delay of `AV_AL_Act` |
| 28 - 31 | Not used |

### See also

AV modes (Page 707)

AV functions (Page 707)

AV error handling (Page 709)

AV messaging (Page 710)

AV block diagram (Page 714)

## 5.2.2 AV modes

### AV operating modes

The block does not have any of its own operating modes.

If the interconnected technologic block is set to the "Out of service" operating mode, the AV block is also set to the "Out of service" mode. In this case `AV_Out` = `AV`.

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit5` =0 and `Feature.Bit5` =1 is as shown below:

| Output Parameters | **`Feature.Bit5` =0** | **`Feature.Bit5` =1** |
|---|---|---|
| AV_Tech.ST | 16#80 | 16#60 |
| AV_AH_Act.ST | 16#XX | 16#60 |
| AV_WH_Act.ST | 16#XX | 16#60 |
| AV_TH_Act.ST | 16#XX | 16#60 |
| AV_TL_Act.ST | 16#XX | 16#60 |
| AV_WL_Act.ST | 16#XX | 16#60 |
| AV_Out.ST | 16#80 | 16#60 |
| AV_AL_Act.ST | 16#XX | 16#60 |

### See also

Description of AV (Page 705)

AV functions (Page 707)

AV error handling (Page 709)

AV messaging (Page 710)

AV I/Os (Page 711)

AV block diagram (Page 714)

## 5.2.3 AV functions

### Functions of AV

The functions for this block are listed below.

### Alarm delays with two time values per limit pair

This block includes the standard function alarm delay for Two time values per limit pair (Page 227).

### Limit monitoring of an additional analog value

This block provides the standard function Limit monitoring of an additional analog value (Page 98).

### Limit monitoring with hysteresis

This block provides the standard function Limit monitoring with hysteresis (Page 104). It is performed via the input parameter `AV_Hyst`.

### Group display SumMsgAct for limit monitoring, CSF and ExtMsgx

The block provides the standard function Group display for limit monitoring, CSF and ExtMsgx (Page 92).

### Forming the signal status for blocks

The worst signal status `ST_Worst` for the block is formed from the following parameter:

- `AV_Out.ST`

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

The simulation in the block AV is activated at the technological block (`SimOn` = 1). The simulation value `SimAV` for the block AV is also specified there. If internal simulation of the technological block is switched on, the output `AV_Out` has the value of `SimAV` and the status 16#60.

### Release for maintenance

The release for maintenance in the AV block is activated at the technological block (`MS_Release` = 1).

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

### Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 230).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) chapter. The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 3 | Disable limit act output when limit value message is disabled (Page 209) |
| 5 | Changes signal status of outputs in OOS (Page 208) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) (transferred to AV via AV_Tech.Mode.Bit) |
| 29 | Signaling limit violation (Page 181) (transferred to AV via AV_Tech.Mode.Bit) |

### See also

AV modes (Page 707)

AV error handling (Page 709)

AV messaging (Page 710)

AV I/Os (Page 711)

AV block diagram (Page 714)

Description of AV (Page 705)

## 5.2.4 AV error handling

### Error handling of AV

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Mode switchover error

### Overview of error numbers

The `ErrorNum` I/O can be used to output various error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; block is not processed |
| 0 | There is no error. |
| 30 | The value of `AV` can no longer be displayed in the REAL number field. |

### Mode switchover error

This error can be output by the block, see the section Error handling (Page 128).

### See also

Description of AV (Page 705)
AV modes (Page 707)
AV functions (Page 707)
AV messaging (Page 710)
AV I/Os (Page 711)
AV block diagram (Page 714)

## 5.2.5 AV messaging

### Messaging

The following messages can be generated for this block:

- Process messages

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId1` | SIG 1 | Alarm - high | $$BlockComment$$<br>`AV` - high alarm limit violated |
| | SIG 2 | Warning - high | $$BlockComment$$<br>`AV` - high warning limit violated |
| | SIG 3 | Tolerance - high | $$BlockComment$$<br>`AV` - high tolerance limit violated |
| | SIG 4 | Tolerance - low | $$BlockComment$$<br>`AV` - low tolerance limit violated |
| | SIG 5 | Warning - low | $$BlockComment$$<br>AV - Low warning limit violated |
| | SIG 6 | Alarm - low | $$BlockComment$$<br>`AV` - low alarm limit violated |

Explanation

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance `MsgEvId1`

| Associated value | Block parameters |
|---|---|
| 1 | Reserved |
| 2 | Reserved |
| 3 | Reserved |
| 4 | `AV_Out` |
| 5 | `ExtVa105` |
| 6 | `ExtVa106` |
| 7 | `ExtVa107` |
| 8 | Reserved |
| 9 | Reserved |
| 10 | Reserved |

The associated values 5 ... 7 are allocated to the parameters `ExtVa105` ... `ExtVa107` and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of AV (Page 705)
AV modes (Page 707)
AV functions (Page 707)
AV error handling (Page 709)
AV I/Os (Page 711)
AV block diagram (Page 714)

## 5.2.6 AV I/Os

### I/Os of AV

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `AV` | Analog value | STRUCT<br>• Value:REAL<br>• ST:BYTE | -<br>• 0.0<br>• 16#80 |
| `AV_Unit` | Unit of measure for the analog value | INT | 1001 |
| `AV_A_DC*` | Delay time for incoming alarms [s] | REAL | 0.0 |
| `AV_A_DG*` | Delay time for outgoing alarms [s] | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| AV_AH_En | 1 = Enable high alarm | BOOL | 1 |
| AV_AH_MsgEn | 1 = Enable high alarm message | BOOL | 1 |
| AV_AL_En | 1 = Enable low alarm | BOOL | 1 |
| AV_AL_MsgEn | 1 = Enable low alarm message | BOOL | 1 |
| AV_OpScale | Limit for scale in bar graph of faceplate for the analog value | STRUCT<br>• High:REAL<br>• Low:REAL | -<br>• 100.0<br>• 0.0 |
| AV_T_DC* | Delay time for incoming tolerances [s] | REAL | 0.0 |
| AV_T_DG* | Delay time for outgoing tolerances [s] | REAL | 0.0 |
| AV_TH_En | 1 = Enable high tolerance | BOOL | 0 |
| AV_TH_MsgEn | 1 = Enable high tolerance message | BOOL | 1 |
| AV_TL_En | 1 = Enable low tolerance | BOOL | 0 |
| AV_TL_MsgEn | 1 = Enable low tolerance message | BOOL | 1 |
| AV_W_DC* | Delay time for incoming warnings [s] | REAL | 0.0 |
| AV_W_DG* | Delay time for outgoing warnings [s] | REAL | 0.0 |
| AV_WH_En | 1 = Enable high warning | BOOL | 1 |
| AV_WH_MsgEn | 1 = Enable high warning message | BOOL | 1 |
| AV_WL_En | 1 = Enable low warning | BOOL | 1 |
| AV_WL_MsgEn | 1 = Enable low warning message | BOOL | 1 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| ExtVa105 | Associated value 5 for messages (MsgEvID1) | ANY | |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| Feature | I/O for additional functions (Page 707) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AV_AH_Act | 1 = High alarm active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AV_AL_Act | 1 = Low alarm active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AV_Out | Analog value output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| AV_Tech | Output parameter with which the input parameter AV of the technologic block has to be connected. This includes tags which are passed on to the technologic block for additional processing. | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#00 |
| AV_TH_Act | 1 = High tolerance active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AV_TL_Act | 1 = Low tolerance active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AV_WH_Act | 1 = High warning active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AV_WL_Act | 1 = Low warning active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see AV error handling (Page 709) | INT | -1 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message acknowledgement status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | Alarm error 1 (output ERROR of first ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 705) | DWORD | 16#00000000 |

### See also

AV modes (Page 707)

AV messaging (Page 710)

AV block diagram (Page 714)

## 5.2.7 AV block diagram

### AV block diagram

A block diagram is not provided for this block.

### See also

Description of AV (Page 705)

AV modes (Page 707)

AV functions (Page 707)

AV error handling (Page 709)

AV messaging (Page 710)

AV I/Os (Page 711)

# 5.3 MonAnL - Monitoring of an analog process tag (Large)

## 5.3.1 Description of MonAnL

### Object name (type+number) and family

Type + number: FB 1845

Family: Monitor

### Area of application for MonAnL

The block is used for the following fields of applications:

- Monitoring an analog process value
- Monitoring of the gradient of an analog process value

**Note**

This block is also available as a small block. A comparison of the MonAnL and MonAnS blocks is available in the section: MonAnL compared to MonAnS (Page 701)

### How it works

The MonAnL block is used to monitor an analog process tag and the corresponding limits. It also monitors the gradient of these signals. The block generates and outputs corresponding messages if limits are violated or if a signal gradient does not meet requirements.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

For the MonAnL block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Monitoring an analog process tag (AnalogMonitoring) (Page 2854)
- Monitoring of an analog process tag for PA/FF devices (AnalogMonitoring_Fb) (Page 2855)

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

The output parameters PV_HysOut, PV_AH_Out, PV_WH_Out, PV_TH_Out, PV_AL_Out, PV_WL_Out and PV_TL_Out are written by the corresponding input parameters PV_Hys, PV_AH_Lim, PV_WH_Lim, PV_TH_Lim, PV_AL_Lim, PV_WL_Lim and PV_TL_Lim.

### Status word allocation for Status1 parameter

You can find a description for each parameter in section MonAnL I/Os (Page 729).

| Status bit | Parameter |
|---|---|
| 0 | Occupied |
| 1 | BatchEn |
| 2 | SimOn |
| 3 | OosAct.Value |
| 4 | OosLi.Value |
| 5 | Not used |
| 6 | OnAct.Value |
| 7 | BypassAct.Value |
| 8 | Display of BypassAct.Value in faceplate (display and operator controls) and block icon |
| 9 | Not used |
| 10 | SimLiOp.Value |
| 11 | Delay of the PV_AH_Lim message |
| 12 | Delay of the PV_WH_Lim message |
| 13 | Delay of the PV_TH_Lim message |
| 14 | Delay of the PV_TL_Lim message |
| 15 | Delay of the PV_WL_Lim message |
| 16 | Delay of the PV_AL_Lim message |
| 17 | Collection of message delays |
| 18 - 29 | Not used |
| 30 | Auxiliary value 1 visible |
| 31 | Auxiliary value 2 visible |

### Status word allocation for Status2 parameter

| Status bit | Parameter |
|---|---|
| 0 | MsgLock |
| 1 | PV_AH_Act.Value |
| 2 | PV_WH_Act.Value |
| 3 | PV_TH_Act.Value |
| 4 | PV_TL_Act.Value |
| 5 | PV_WL_Act.Value |
| 6 | PV_AL_Act.Value |
| 7 | PV_AH_En OR Feature.Bit23 |
| 8 | PV_WH_En OR Feature.Bit23 |

| Status bit | Parameter |
|---|---|
| 9 | PV_TH_En OR Feature.Bit23 |
| 10 | PV_TL_En OR Feature.Bit23 |
| 11 | PV_WL_En OR Feature.Bit23 |
| 12 | PV_AL_En OR Feature.Bit23 |
| 13 | PV_AH_MsgEn |
| 14 | PV_WH_MsgEn |
| 15 | PV_TH_MsgEn |
| 16 | PV_TL_MsgEn |
| 17 | PV_WL_MsgEn |
| 18 | PV_AL_MsgEn |
| 19 | GradHUpAct.Value |
| 20 | GradHDnAct.Value |
| 21 | GradLAct.Value |
| 22 | GradHUpEn |
| 23 | GradHDnEn |
| 24 | GradLEn |
| 25 | GradHUpMsgEn |
| 26 | GradHDnMsgEn |
| 27 | GradLMsgEn |
| 28 | 0 = falling measured value<br>1 = rising measured value |
| 29 | 1= Suppressed messages |
| 30 | Internal error |
| 31 | MS_RelOp |

### Status word allocation for Status3 parameter

| Status bit | Parameter |
|---|---|
| 0 | Effective signal 1 of the message block connected via EventTsIn |
| 1 | Effective signal 2 of the message block connected via EventTsIn |
| 2 | Effective signal 3 of the message block connected via EventTsIn |
| 3 | Effective signal 4 of the message block connected via EventTsIn |
| 4 | Effective signal 5 of the message block connected via EventTsIn |
| 5 | Effective signal 6 of the message block connected via EventTsIn |
| 6 | Effective signal 7 of the message block connected via EventTsIn |
| 7 | Effective signal 8 of the message block connected via EventTsIn |
| 8 - 15 | Effective signal 9..16 of the message block connected via EventTsIn |
| 16 - 31 | Not used |

### See also

MonAnL functions (Page 719)

MonAnL messaging (Page 727)

MonAnL block diagram (Page 738)

MonAnL error handling (Page 726)

MonAnL modes (Page 718)

## 5.3.2 MonAnL modes

### MonAnL operating modes

The block can be operated using the following modes:

- On (Page 77)
- Out of service (Page 77)

### "On"

You can find general information about the "On" mode in the On (Page 77) section.

### "Out of service"

You can find general information about the "Out of service" mode in the Out of service (Page 77) section.

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit5` =0 and `Feature.Bit5` =1 is as shown below:

| Output Parameters | **`Feature.Bit5` =0** | **`Feature.Bit5` =1** |
|---|---|---|
| PV_Out.ST | 16#80 | 16#60 |
| PV_Grad.ST | 16#80 | 16#60 |
| PV_GradPP.ST | 16#80 | 16#60 |
| PV_GradNP.ST | 16#80 | 16#60 |
| PV_AH_Act.ST | 16#XX | 16#60 |
| PV_WH_Act.ST | 16#XX | 16#60 |
| PV_TH_Act.ST | 16#XX | 16#60 |
| PV_TL_Act.ST | 16#XX | 16#60 |
| PV_WL_Act.ST | 16#XX | 16#60 |
| PV_AL_Act.ST | 16#XX | 16#60 |
| GradHUpAct.ST | 16#XX | 16#60 |
| GradHDnAct.ST | 16#XX | 16#60 |
| GradLAct.ST | 16#XX | 16#60 |

**See also**

MonAnL block diagram (Page 738)
MonAnL I/Os (Page 729)
MonAnL messaging (Page 727)
MonAnL error handling (Page 726)
MonAnL functions (Page 719)
Description of MonAnL (Page 715)

## 5.3.3 MonAnL functions

### Functions of MonAnL

The functions for this block are listed below.

### Alarm delays with two time values

With the Separate delay times for each alarm (Page 181) function (`Feature.Bit8`), separate input parameters can be set for the alarm delay on the high and low limits.

`Feature.Bit8` = 0

This block includes the standard function alarm delay for Two time values per limit pair (Page 227).

`Feature.Bit8` = 1

This block includes the standard function alarm delay for Two time values for each individual limit (Page 228).

### Limit monitoring of the process value

This block provides the standard function Limit monitoring of the process value (Page 93).

### Group display SumMsgAct for limit monitoring, CSF and ExtMsgx

The block provides the standard function Group display for limit monitoring, CSF and ExtMsgx (Page 92).

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Gradient monitoring

The PV_Grad gradient is calculated with a delay function, LagTime. This setting smoothes the jumps of the PV input value with gradient calculation.

The gradient peak values are output at the output parameters PV_GradNP (negative slope) and PV_GradPP (positive slope). They are reset as soon as the operator issues the reset command.

Thegradient PV_Grad gradient can be monitored for the following limits:

- Limit (high) for positive gradients (GradHUpLim)
- Limit (high) for negative gradients (GradHDnLim)
- Limit (low) for absolute gradients (GradLLim)

The individual monitoring functions are activated at the corresponding "Enable" parameters, e.g. GradHUpEn for activating the high gradient limit for positive gradients (GradHUpLim).

The gradient PV_Grad of the process value is calculated in units per second, per minute, or per hour, depending on the configuration of the TimeFactor:

- TimeFactor = 0: The unit of gradient limitation is Unit/Second

- TimeFactor = 1: The unit of gradient limitation is Unit/Minute

- TimeFactor = 2: Unit of gradient limitation is Unit/hour

The limits GradHUpLim, GradHDnLim and GradLLim must be entered with positive values in units per second, per minute or per hour, according to the configuration of the TimeFactor.

Reaching or exceeding the limits can be indicated and signaled at the corresponding outputs:

GradLAct: -GradLLim <= PV_Grad <= GradLLim

GradHupAct: PV_Grad is positive: PV_Grad >= GradHUpLim

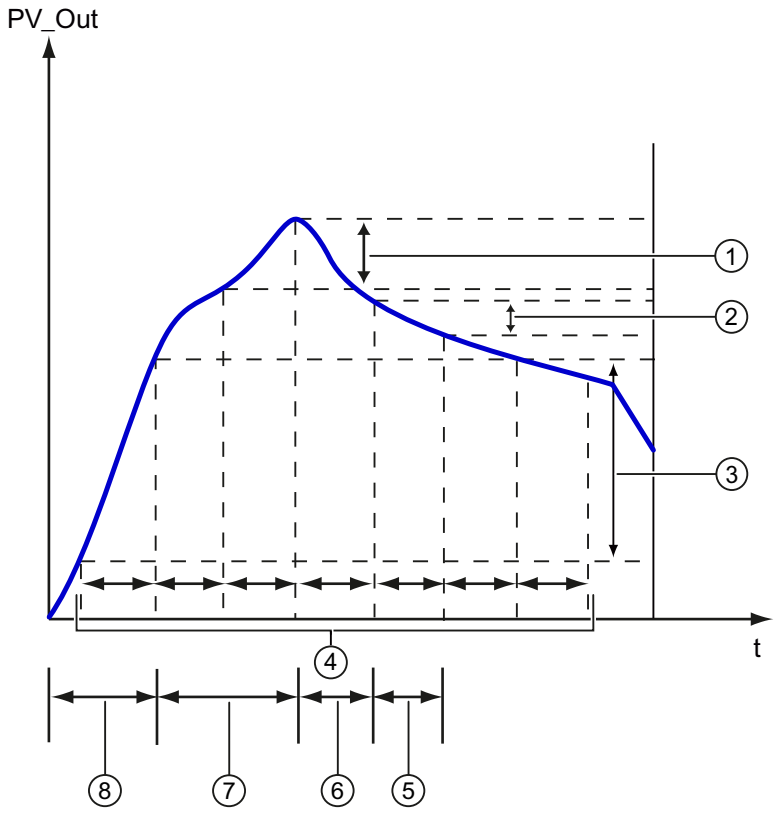
GradHDnAct: PV_Grad is negative: PV_Grad <= -GradHDnLim

Messages can be output for these alarms. You activate these as follows:

- Message for alarms (high) for positive gradients: GradHUpMsgEn = 1
- Message for alarms (high) for negative gradients: GradHDnMsgEn = 1
- Message for alarms (low) for absolute gradients: GradLMsgEn = 1

## Example of generating alarms for gradient monitoring

The following example shows how alarms for gradient monitoring are generated.




| Number | Description |
|---|---|
| 1 | Gradient is negative and `PV_Grad ≤ -GradHDnLim` |
| 2 | Gradient `ABS(PV_Grad) ≤ GradLLim` |
| 3 | Gradient is positive and `PV_Grad ≥ GradHUpLim` |
| 4 | Sampling time (`SampleTime`) |
| 5 | Alarm (low) for absolute gradients |
| 6 | Alarm (high) for negative gradients |
| 7 | No alarm |
| 8 | Alarm (high) for positive gradients |

In every cycle, a value of `PV_Grad` is calculated. When this value reached the limits, the alarms are generated.

## Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 239).

### Providing PV limit at the output

The following input parameters are also available at the output for making additional interconnections to other blocks:

| | |
|---|---|
| • PV_HysOut | := PV_Hyst |
| • PV_AH_Out | := PV_AH_Lim |
| • PV_WH_Out | := PV_WH_Lim |
| • PV_TH_Out | := PV_TH_Lim |
| • PV_TL_Out | := PV_TL_Lim |
| • PV_WL_Out | := PV_WL_Lim |
| • PV_AL_Out | := PV_AL_Lim |

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `PV_Out.ST`

### Dead band

To suppress values that fluctuate around the zero point, this block has the standard function Dead band (Page 67).

### Smoothing PV

The block smoothes the input value `PV` using a first order time delay. This delay time can be configured with the parameter `SmoothTi`. The input of `SmoothTi` is limited to 0 to 999999 sec. The block writes back the limits if the input value is outside the limits.

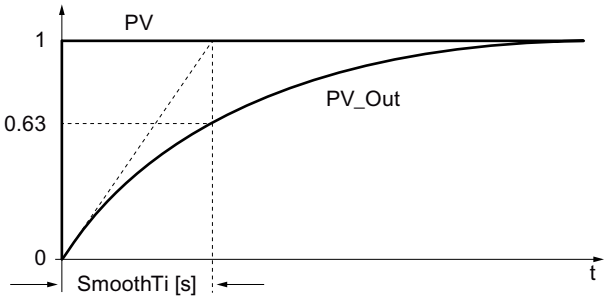
The block works according to the following formula:

$PV_Out_{(n)} = PV_{(n)} + ( PV_Out_{(n-1)} - PV_{(n)} ) * e^{(-SampleTime/SmoothTi)}$

Where:

- `PV_Out` = Output value
- `SmoothTi` = Smooth time
- `SampleTime` = Sampling time
- `PV` = Input value

The formula is valid only if `SmoothTi` > 0. If `SmoothTi` = 0, the input is passed directly to the output. If the input value is outside the REAL range limits, the calculation is stopped. If the input value is inside the range limits again, the calculation is resumed automatically.




### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Process value (`SimPV`, `SimPV_Li`)

### Bypass function

This block provides the standard function Bypassing signals (Page 115).

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

### Disabling of limits

The block provides the standard function Limits disabled in faceplate (Page 206).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 3 | Disable limit act output when limit value message is disabled (Page 209) |
| 5 | Changes signal status of outputs in OOS (Page 208) |
| 8 | Separate delay times for each alarm (Page 181) |
| 9 | Substitution value is active if the block is in bypass (Page 199) |
| 11 | Suppress MsgLock and “Out of service” mode for a connected message block (Page 202) |

| Bit | Function |
|---|---|
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 23 | Limits disabled in faceplate (Page 206) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 28 | Disabling operating points (Page 156) |
| 29 | Signaling limit violation (Page 181) |

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can activate bypass functionality |
| 5 | 1 = Operator can deactivate bypass functionality |
| 6 | 1 = Operator can change the simulation value `SimPV` |
| 7 | 1 = Operator can reset the maximum values |
| 8 | 1 = Operator can activate / deactivate messages via `GradHUpMsgEn` |
| 9 | 1 = Operator can activate / deactivate messages via `GradHDnMsgEn` |
| 10 | 1 = Operator can activate / deactivate messages via `GradLMsgEn` |
| 11 | 1 = Operator can activate the Simulation function |
| 12 | 1 = Operator can activate the Release for maintenance function |
| 13 | 1 = Operator can change the limit (`PV`) for high alarm |
| 14 | 1 = Operator can change the limit (`PV`) for high warning |
| 15 | 1 = Operator can change the limit (`PV`) for high tolerance |
| 16 | 1 = Operator can change the limit (`PV`) for hysteresis |
| 17 | 1 = Operator can change the limit (`PV`) for low alarm |
| 18 | 1 = Operator can change the limit (`PV`) for low warning |
| 19 | 1 = Operator can change the limit (`PV`) for low tolerance |
| 20 | 1 = Operator can change the value for the high gradient limit for positive slopes (`GradHUpLim`) |
| 21 | 1 = Operator can change the value for high gradient limit for negative slopes (`GradHDnLim`) |
| 22 | 1 = Operator can change the value for the low gradient limit for positive and negative slopes (`GradLLim`) |
| 23 | 1 = Operator can change the dead band parameter `DeadBand` |
| 24 | 1 = Operator can activate / deactivate messages via `PV_AH_MsgEn` |
| 25 | 1 = Operator can activate / deactivate messages via `PV_WH_MsgEn` |
| 26 | 1 = Operator can activate / deactivate messages via `PV_TH_MsgEn` |

| Bit | Function |
|---|---|
| 27 | 1 = Operator can activate / deactivate messages via PV_TL_MsgEn |
| 28 | 1 = Operator can activate / deactivate messages via PV_WL_MsgEn |
| 29 | 1 = Operator can activate / deactivate messages via PV_AL_MsgEn |
| 30 | 1 = Operator can enter SmoothTi |
| 31 | Reserved |

The block has the following permissions for the OS1_Perm parameter:

| Bit | Function |
|---|---|
| 0 - 15 | reserved |

**Note**

If you interconnect a parameter that is also listed in OS_Perm as a parameter, you have to reset the corresponding OS_Perm bit.

### Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 230).

### Specifying the display area for process and setpoint values as well as operations

This block provides the standard function Display and operator input area for process values and setpoints (Page 233).

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of MonAnL (Page 715)

MonAnL messaging (Page 727)

MonAnL I/Os (Page 729)

MonAnL block diagram (Page 738)

MonAnL error handling (Page 726)

MonAnL modes (Page 718)

EventTs functions (Page 2046)

Error handling (Page 128)

Sealing the valve (Page 187)

## 5.3.4 MonAnL error handling

### Error handling of MonAnL

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following error message can be generated at this block:

- Error numbers
- Control system fault (CSF)

### Overview of error numbers

The ErrorNum I/O can be used to output various error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed. |
| 0 | There is no error. |
| 11 | SmoothTi < 0 or SmoothTi > 999999:<br>Due to the internal limitation to 0 or 999999 by the block, the error number has the value 11 only for one cycle. If SmoothTi is connected and is written in every cycle, the value 11 will be retained. |
| 30 | The value of PV can no longer be displayed in the REAL number field. |
| 43 | TimeFactor < 0 or TimeFactor > 2 |

### Control system fault (CSF)

An external signal can be activated via the CSF input. If this signal changes to = 1, a control system fault is triggered. Refer to the Error handling (Page 128) section for more on this.

### See also

MonAnL block diagram (Page 738)

MonAnL I/Os (Page 729)

MonAnL messaging (Page 727)

MonAnL functions (Page 719)

MonAnL modes (Page 718)
Description of MonAnL (Page 715)

## 5.3.5 MonAnL messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Process messages
- Instance-specific messages

### Process control fault

You can interconnect an external fault (signal) to input parameter `CSF`. If it changes to `CSF` = 1, a process control fault is triggered (`MsgEvId2`, SIG 2).

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId1` | SIG 1 | Alarm - high | $$BlockComment$$<br>`PV` - high alarm limit violated |
| | SIG 2 | Warning - high | $$BlockComment$$<br>`PV` - high warning limit violated |
| | SIG 3 | Tolerance - high | $$BlockComment$$<br>`PV` - high tolerance limit violated |
| | SIG 4 | Tolerance - low | $$BlockComment$$<br>`PV` - low tolerance limit violated |
| | SIG 5 | Warning - low | $$BlockComment$$<br>`PV` - low warning limit violated |
| | SIG 6 | Alarm - low | $$BlockComment$$<br>`PV` - low alarm limit violated |
| | SIG 7 | Alarm - high | $$BlockComment$$<br>Limit (high) for positive gradients violated |
| | SIG 8 | Alarm - high | $$BlockComment$$<br>Limit (high) for negative gradients violated |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Instance-specific messages

You can use up to three instance-specific messages with this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId2 | SIG 1 | Alarm - low | $$BlockComment$$<br>Limit (low) for absolute gradients violated |
| | SIG 2 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |
| | SIG 3 | AS process control message - fault | $$BlockComment$$<br>External message 1 |
| | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External message 2 |
| | SIG 5 | AS process control message - fault | $$BlockComment$$<br>External message 3 |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance **MsgEvId1**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |
| 4 | ExtVa104 |
| 5 | ExtVa105 |
| 6 | ExtVa106 |
| 7 | ExtVa107 |
| 8 | ExtVa108 |
| 9 | PV_Out.Value |
| 10 | Reserved |

The associated values 4 to 9 are allocated to the parameters ExtVa104...ExtVa108, PV_Out.Value and can be used.

Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### Associated values for message instance **MsgEvId2**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |
| 4 | ExtVa204 |

| Associated value | Block parameters |
|---|---|
| 5 | ExtVa205 |
| 6 | ExtVa206 |
| 7 | ExtVa207 |
| 8 | ExtVa208 |
| 9 | PV_Out.Value |
| 10 | Reserved |

The associated values 4 to 9 are allocated to the parameters ExtVa204...ExtVa208, PV_Out.Value and can be used.

Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of MonAnL (Page 715)

MonAnL functions (Page 719)

MonAnL I/Os (Page 729)

MonAnL block diagram (Page 738)

MonAnL error handling (Page 726)

MonAnL modes (Page 718)

## 5.3.6 MonAnL I/Os

MonAnL I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AddOn | AddOn block connector for device specific view | ANY | - |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkTypeData* | This input can not be edit or linked and is for internal usage only! | | |
| BypLiOp | 1 = Bypass commands via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypLock | 1 = Bypass activation or deactivation is locked for operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypPV | Substitution value if block is in bypass | REAL | 0.0 |
| BypPVLi | 1 = Select bypass PV (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypPVOp | 1 = Select bypass PV (via operator) | BOOL | 0 |
| CSF | 1 = External error (control system fault) Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DeadBand | Width of dead band | REAL | 0.0 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| EventTsIn | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ExtMsg3 | Binary input for freely selectable message 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa104 | Associated value 4 for messages (MsgEvID1) | ANY | |
| ExtVa105 | Associated value 5 for messages (MsgEvID1) | ANY | |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| ExtVa108 | Associated value 8 for messages (MsgEvID1) | ANY | |
| ExtVa204 | Associated value 4 for messages (MsgEvID2) | ANY | |
| ExtVa205 | Associated value 5 for messages (MsgEvID2) | ANY | |
| ExtVa206 | Associated value 6 for messages (MsgEvID2) | ANY | |
| ExtVa207 | Associated value 7 for messages (MsgEvID2) | ANY | |
| ExtVa208 | Associated value 8 for messages (MsgEvID2) | ANY | |
| Feature | I/O for additional functions (Page 719) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| GradHUpLim | Gradient monitoring: Limit (high) for positive gradients | REAL | 10.0 |
| GradHDnLim | Gradient monitoring: Limit (high) for negative gradients | REAL | 10.0 |
| GradLLim | Gradient monitoring: Limit (low) for absolute gradients | REAL | 1.0 |
| GradHUpEn | 1 = Activate gradient monitoring (high) for positive changes | BOOL | 1 |
| GradHDnEn | 1 = Activate gradient monitoring (high) for negative changes | BOOL | 1 |
| GradLEn | 1 = Activate gradient monitoring (low) | BOOL | 0 |
| GradHUpMsgEn | 1 = Activate gradient (high) message for positive changes | BOOL | 1 |
| GradHDnMsgEn | 1 = Activate gradient (high) message for negative changes | BOOL | 1 |
| GradLMsgEn | 1 = Activate gradient (low) message | BOOL | 1 |
| LagTime | Delay time [s] | REAL | 1.0 |
| MS_RelOp* | 1 = Release for maintenance by OS operator | BOOL | 0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgEvId2 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 719) | STRUCT<br>• Bit 0: BOOL<br>• Bit 10: BOOL<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| OS1Perm | I/O for operator permissions (Page 719) | STRUCT<br>• Bit 0: BOOL<br>• Bit 15: BOOL | -<br>• 1<br>• 1 |
| PV* | Process value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_A_DC* | Delay time for incoming PV alarms for overshoot/undershoot or only for undershot process values [s] | REAL | 0.0 |
| PV_A_DG* | Delay time for outgoing PV alarms for overshoot/undershoot or only for undershot process values [s] | REAL | 0.0 |
| PV_AH_DC* | Delay time for incoming alarms for overshot process values [s] | REAL | 0.0 |
| PV_AH_DG* | Delay time for outgoing alarms for overshot process values [s] | REAL | 0.0 |
| PV_AH_En | 1 = Enable PV alarm limit (high) | BOOL | 1 |
| PV_AH_Lim | Limit PV alarm (high) | REAL | 95.0 |
| PV_AH_MsgEn | 1 = Enable PV alarm (high) message | BOOL | 1 |
| PV_AL_En | 1 = Enable PV alarm limit (low) | BOOL | 1 |
| PV_AL_Lim | PV alarm limit (low) | REAL | 5.0 |
| PV_AL_MsgEn | 1 = Enable PV alarm (low) message | BOOL | 1 |
| PV_Hyst* | Hysteresis for PV alarm, warning and tolerance limits | REAL | 0.0 |
| PV_OpScale | Limit for scale in PV bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| PV_T_DC* | Delay time for incoming PV tolerance messages for overshoot/undershoot or only for undershot process values [s] | REAL | 0.0 |
| PV_T_DG* | Delay time for outgoing PV tolerance messages for overshoot/undershoot or only for undershot process values [s] | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_TH_DC* | Delay time for incoming tolerance messages for overshot process values [s] | REAL | 0.0 |
| PV_TH_DG* | Delay time for outgoing tolerance messages for overshot process values [s] | REAL | 0.0 |
| PV_TH_En | 1 = Enable PV tolerance limit (high) | BOOL | 0 |
| PV_TH_Lim | Limit PV tolerance message (high) | REAL | 85.0 |
| PV_TH_MsgEn | 1 = Enable message for PV tolerance message (high) | BOOL | 1 |
| PV_TL_En | 1 = Enable PV tolerance limit (low) | BOOL | 0 |
| PV_TL_Lim | Limit PV tolerance message (low) | REAL | 15.0 |
| PV_TL_MsgEn | 1 = Activate message for PV tolerance message (low) | BOOL | 1 |
| PV_Unit | Unit of measure for process value | INT | 1001 |
| PV_W_DC* | Delay time for incoming PV warnings for overshoot/undershoot or only for undershot process values [s] | REAL | 0.0 |
| PV_W_DG* | Delay time for outgoing PV warnings for overshoot/undershoot or only for undershot process values [s] | REAL | 0.0 |
| PV_WH_DC* | Delay time for incoming warnings for overshot process values [s] | REAL | 0.0 |
| PV_WH_DG* | Delay time for outgoing warnings for overshot process values [s] | REAL | 0.0 |
| PV_WH_En | 1 = Enable PV warning limit (high) | BOOL | 1 |
| PV_WH_Lim | Limit PV warning (high) | REAL | 90.0 |
| PV_WH_MsgEn | 1 = Enable PV warning (high) message | BOOL | 1 |
| PV_WL_En | 1 = Enable PV warning limit (low) | BOOL | 1 |
| PV_WL_Lim | Limit PV warning (low) | REAL | 10.0 |
| PV_WL_MsgEn | 1 = Enable PV warning (low) message | BOOL | 1 |
| RstBypLi | 1 = Reset bypass PV (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstBypOp | 1 = Reset bypass PV (via operator) | BOOL | 0 |
| RstOp* | 1 = Operator reset of the gradient's peak values | BOOL | 0 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | 1 = Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SelFp2 | 1 = Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | - |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SimPV* | Process value that is used for SimOn = 1 | REAL | 0.0 |
| SimPV_Li | Process value that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| SmoothTi* | Smooth time [s] (limited to 0 to 999999) | REAL | 0.0 |
| TimeFactor | Time unit:<br>0 = Seconds<br>1 = Minutes<br>2 = Hours | INT | 0 |
| UA1unit | Unit of measure for analog auxiliary value 1 | INT | 0 |
| UA2unit | Unit of measure for analog auxiliary value 2 | INT | 0 |
| UserAna1 | Analog auxiliary value 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UserAna2 | Analog auxiliary value 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| BypassAct | 1 = Bypass is activated in this block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see MonAnL error handling (Page 726) | INT | -1 |
| GradHUpAct | 1 = Gradient alarm (high) for positive changes.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| GradHDnAct | 1 = Gradient alarm (high) for negative changes.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| GradLAct | 1 = Low gradient alarm.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message acknowledgement status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |
| MsgAckn2 | Message acknowledgement status 2 (output ACK_STATE of second ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | 1 = Alarm error 1 (output ERROR of the first ALARM_8P) | BOOL | 0 |
| MsgErr2 | 1 = Alarm error 2 (output ERROR of the second ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| MsgStat2 | Message status 2 (output STATUS of second ALARM_8P) | WORD | 16#0000 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feed-forwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1_Perm | WORD | 16#FFFF |
| OS1PermLog | Display of OS1_Perm with settings changed by the block algorithm | WORD | 16#FFFF |
| PV_AH_Act | 1 = PV alarm (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_AH_Out | PV - alarm limit (high) output | REAL | 0.0 |
| PV_AL_Act | 1 = PV alarm (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_AL_Out | PV - alarm limit (low) output | REAL | 0.0 |
| PV_Grad | Gradient value. | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_GradPP | Gradient maximum peak value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_GradNP | Gradient minimum peak value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_HysOut | Hysteresis for PV alarm, output | REAL | 0.0 |
| PV_OpScaleOut | PV_OpScale as output | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| PV_Out | Output for process value | STRUCT<br>• Value: REAL<br>• ST: BYTE<br>• Bit 1: BOOL of internal fill byte (Bypass information bit) | -<br>• 0.0<br>• 16#80<br>• 0 |
| PV_UnitOut | PV_Unit as output | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_TH_Act | 1 = PV tolerance message (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_TH_Out | PV tolerance limit (high) output | REAL | 0.0 |
| PV_TL_Act | 1 = PV tolerance message (low) active. You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_TL_Out | PV tolerance limit (low) output | REAL | 0.0 |
| PV_WH_Act | 1 = PV warning (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_WH_Out | Limit PV warning (high) output | REAL | 0.0 |
| PV_WL_Act | 1 = PV warning (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_WL_Out | Limit PV warning (low) output | REAL | 0.0 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 715) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 715) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 715) | DWORD | 16#00000000 |
| SumMsgAct | 1 = Group message is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| UA1unitOut | UA1 as output | INT | 0 |
| UA2unitOut | UA2 as output | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| `UserAna1Out` | `UserAna1` as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| `UserAna2Out` | `UserAna2` as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |

**See also**

MonAnL messaging (Page 727)
MonAnL block diagram (Page 738)
MonAnL modes (Page 718)

## 5.3.7 MonAnL block diagram

### MonAnL block diagram

A block diagram is not provided for this block.

**See also**

MonAnL I/Os (Page 729)
MonAnL messaging (Page 727)
MonAnL error handling (Page 726)
MonAnL functions (Page 719)
MonAnL modes (Page 718)
Description of MonAnL (Page 715)

## 5.3.8 Operator control and monitoring

### 5.3.8.1 MonAnL views

#### Views of the MonAnL block

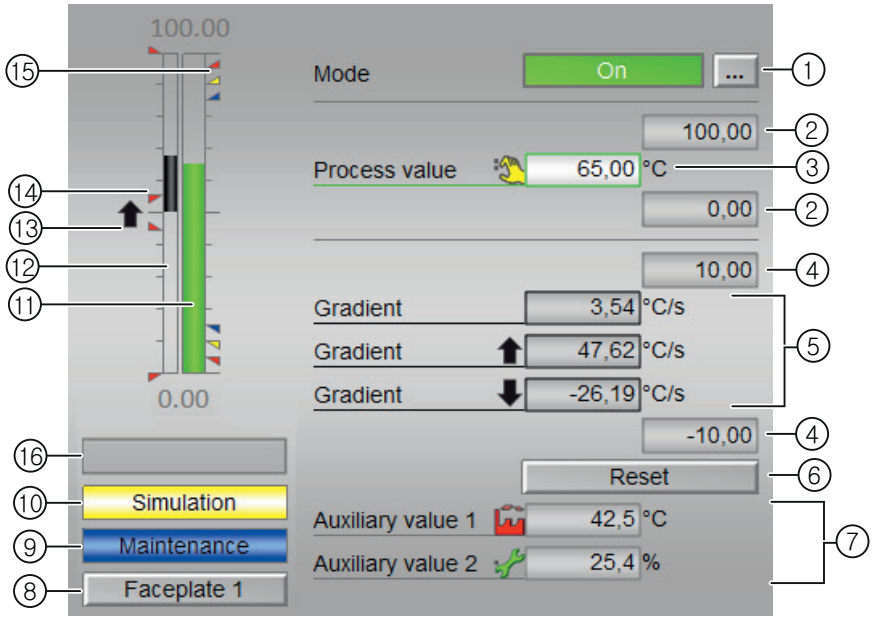
The block MonAnL provides the following views:

- MonAnL standard view (Page 740)
- Alarm view (Page 336)
- MonAnL limit view (Page 743)
- Trend view (Page 341)
- MonAnL parameter view (Page 745)
- MonAnL preview (Page 746)
- MonAnL AddOn view (Page 747)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for MonAnL (Page 748)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 5.3.8.2 MonAnL standard view

#### MonAnL standard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Bar graph (Page 376)
- Out of service (Page 77)

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the operating mode.

#### (2) High and low scale range for the process value

These values provide information on the display range for the bar graph of the process value. The scale range is defined in the engineering system.

#### (3) Display of the process value including signal status

This area shows the current process value with the corresponding signal status.

If text is configured for this command, it is displayed as additional text and button label for command selection. Additional information is available in the section Labeling of buttons and text (Page 235).

You can change the text for the process value with the `PV_Out` parameter.

### (4) High and low scale range for the gradient value

These values provide information on the display range for the bar graph of the gradient. The scale range corresponds to 10% of the scale range for the process value: For example, once you have specified a process value scale range of 0 to 100, the scale range of the gradient will be automatically set to a value between -10 and 10.

The current gradient value is displayed when one of the following monitoring functions is activated:

- Gradient monitoring for positive changes `(GradHUpEn = 1)`
- Gradient monitoring for negative changes `(GradHDnEn = 1)`
- Gradient monitoring `(GradLEn = 1)`

### (5) Display of the gradient

This area shows the current, minimum and maximum gradient value and the rise and fall of the value. This display of the minimum and maximum gradient value functions like a min/max pointer.

The current gradient value is displayed when one of the following monitoring functions is activated:

- Gradient monitoring for positive changes `(GradHUpEn = 1)`
- Gradient monitoring for negative changes `(GradHDnEn = 1)`
- Gradient monitoring `(GradLEn = 1)`

The maximum peak gradient value is displayed when the gradient monitoring is activated for positive changes `(GradHUpEn = 1)`

The minimum peak gradient value is displayed when the gradient monitoring is activated for negative changes `(GradHDnEn = 1)`

### (6) Resetting the peak values of the gradient

You can use this button to reset the maximum or minimum peak value of the gradient (`PV_GradPP` and `PV_GradNP` output parameters).

The button is displayed when gradient monitoring is activated for positive `(GradHUpEn = 1)` or negative changes `(GradHDnEn = 1)`.

### (7) Display of auxiliary values

You can use this area to display two auxiliary values that have been configured in the engineering system. You can find additional information on this in the Displaying auxiliary values (Page 239) section.

### (8) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Opening additional faceplates (Page 234) section.

### (9) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

### (10) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"

### (11) Bar graph for the "process value"

This area shows the current "Process value" in the form of a bar graph. You can also see the high low limits of the bargraph's scale.The visible area in the bar graph depends on the configuration in the engineering system (ES). For more information refer Bar graph (Page 376) section.

### (12) Bar graph for the gradient

This area shows the current gradient value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

The bar graph is displayed when gradient monitoring is activated for positive `(GradHUpEn = 1)` or negative changes `(GradHDnEn = 1)`.

### (13) Display of the gradient

This display indicates the movement of the gradient up (↑) or down (↓).

Gradient monitoring is displayed when the gradient value PV_Grad ≠ 0 and one of the following monitoring functions is activated:

- Gradient monitoring for positive changes `(GradHUpEn = 1)`
- Gradient monitoring for negative changes `(GradHDnEn = 1)`
- Gradient monitoring `(GradLEn = 1)`

### (14) Display of limits in the bar graph

This area shows you the specified limits. Refer to the MonAnL limit view (Page 743) section for more on this.

The display only appears when the bar for the gradients is also displayed.

### (15) Limit display

These colored triangles show you the specified limits in the respective bar graph.

### (16) Display area for block states

This area provides additional information on the operating state of the block:

- Internal error
- Invalid Signal

### See also

On (Page 77)

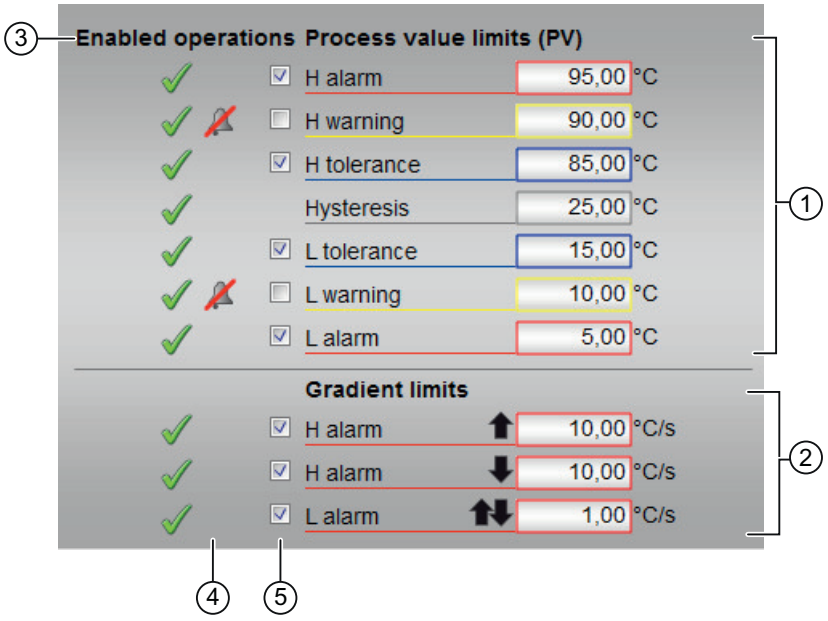
## 5.3.8.3 MonAnL limit view

### Limit view of MonAnL

Several values are set in this view by default:

- Process value limits
- Gradient limits

The toolbars of the faceplate and the block icon indicate when the limits are reached or violated.




### (1) Process value limits

In this area, you can enter the limits for the process value. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H alarm": Alarm high
- "H warning": Warning high

- "H tolerance": Tolerance high
- "Hysteresis"
- "L tolerance": Tolerance low
- "L warning": Warning low
- "L alarm": Alarm low

**Note**

The text displayed are not valid for user-configured message classes. For more information refer Limit operation and display in the faceplate (Page 355).

### (2) Gradient limits

You can enter the gradient limits in this area. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H alarm ⇑": Gradient for the high slope for positive changes
- "H alarm ⇓": Gradient for the high slope for negative changes
- "L alarm ⇑ ⇓": Gradient for the low slope (absolute)

### (3) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

### (4) Message suppression / delays

Message suppression indicates whether or not the suppression of the associated message in the AS block is activated with the `xx_MsgEn` parameter. The output of messages is not suppressed when the block is installed (all `xx_MsgEn` parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

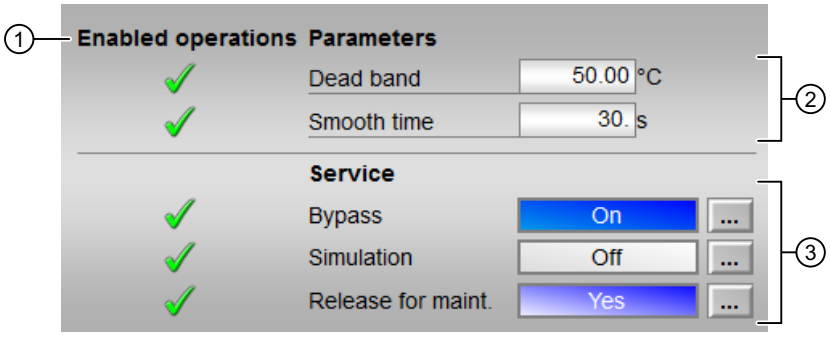
Alarm delays are also displayed in this position; for more on this see section Area of application for alarm delays (Page 225).

### (5) Suppress messages

You can enable / disable messages by setting the check mark.

## 5.3.8.4 MonAnL parameter view

### Parameter view of MonAnL




### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

### (2) Parameter

You can change the following parameter in this area:

- "Dead band"
- "Smooth time"

You can find more information about this in the section Changing values (Page 291).

### (3) Service

You can select the following functions in this area:

- "Bypass"
- "Simulation"
- "Release for maintenance"

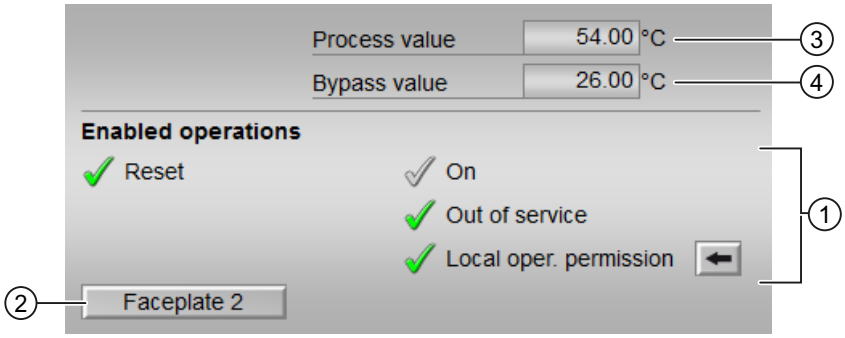
Refer to the Switching operating states and operating modes (Page 289) section for more on this.

You can find information on this area in the section:

- Bypassing signals (Page 115)
- Simulating signals (Page 64)
- Release for maintenance (Page 70)

### 5.3.8.5 MonAnL preview

#### Preview of MonAnL




#### (1) Enable operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- Reset: You can reset the peak value of the gradient.
- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operator permission": Use the ← button to switch to the standard view of the OpStations block. You can find additional information on this in the section Operator control permissions (Page 285) .

### (2) Navigation button for switching to the standard view of any faceplate

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234) .

### (3) Process value

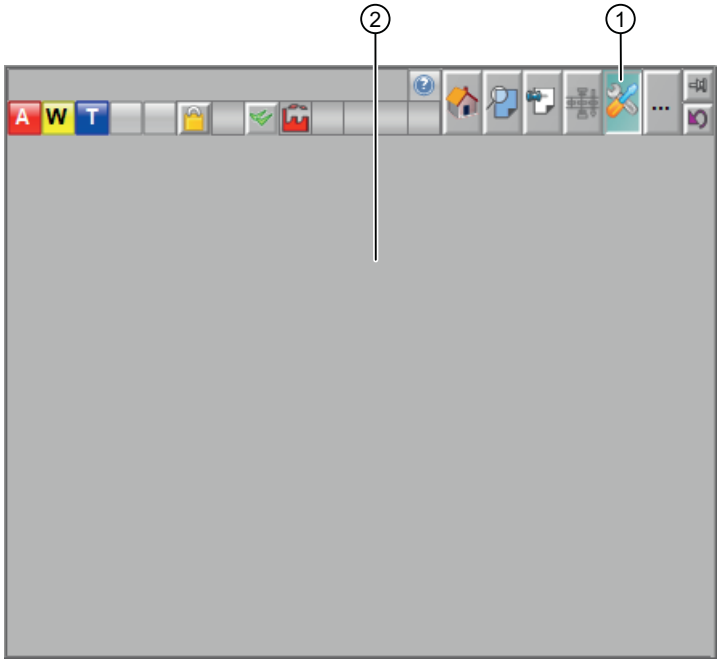
This area displays the real process value (`PV`).

### (4) Bypass value

This area displays the bypass value (`BypPV`).

## 5.3.8.6 MonAnL AddOn view

### MonAnL AddOn view




(1) AddOn view icon

(2) Work space of AddOn view: The view of the faceplate can be customized as per the operator needs.

For more information, on AddOn view, refer to AUTOHOTSPOT

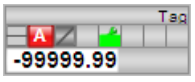
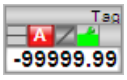
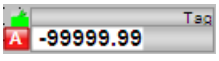
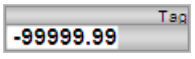
### 5.3.8.7 Block icon for MonAnL

## Block symbols for MonAnL

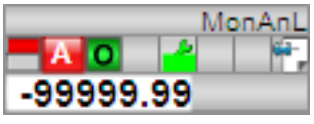
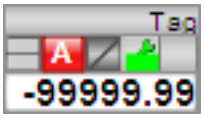
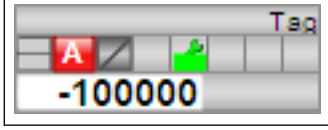
A variety of block symbols are available with the following functions:

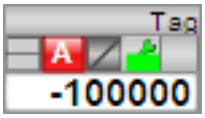
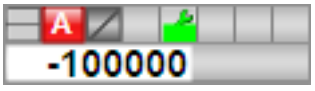
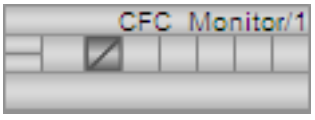
- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as control system faults
- Operating modes
- Signal status, release for maintenance
- Memo display
- Process value

The block symbols from template @TemplateAPLV9.PDL:

| Symbols | Selection of the block symbol in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | |
|  | 4 | |

The block symbols from template @TemplateAPLV7.PDL:

| Symbols | Selection of the block symbol in CFC | Special features |
|---|---|---|
|  | 1 | Block symbol in the full display |
|  | 2 | |
|  | 3 | |

| Symbols | Selection of the block symbol in CFC | Special features |
|---|---|---|
|  | 4 | |
|  | 5 | |
|  | 6 | |
|  | 7 | |
|  | 8 | |
|  | - | Block symbol in "Out of service" mode (example with block symbol type 1) |

Additional information on the block symbol and the control options in the block symbol is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269).

# 5.4 MonAnS - Monitoring of an analog process tag (Small)

## 5.4.1 Description of MonAnS

### Object name (type + number)

Type + number: FB 1912

Family: Monitor

### Area of application for MonAnS

The block is used for the following fields of application

- Monitoring an analog process value

**Note**

This block is also available as a large block. A comparison of the MonAnL and MonAnS blocks is available in the section: MonAnL compared to MonAnS (Page 701)

### How it works

The MonAnS block is used to monitor an analog process tag and the corresponding limits. The block generates and outputs messages if any violation of limits is detected.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section MonAnS I/Os (Page 758).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |

| Status bit | Parameter |
|---|---|
| 3 | OosAct.Value |
| 4 | OosLi.Value |
| 5 | Not used |
| 6 | OnAct.Value |
| 7 - 10 | Not used |
| 11 | Delay of the PV_AH_Lim message |
| 12 | Delay of the PV_WH_Lim message |
| 13 - 14 | Not used |
| 15 | Delay of the PV_WL_Lim message |
| 16 | Delay of the PV_AL_Lim message |
| 17 | Collection of message delays |
| 18 - 31 | Not used |

### Status word allocation for Status2 parameter

| Status bit | Parameter |
|---|---|
| 0 | MsgLock |
| 1 | PV_AH_Act.Value |
| 2 | PV_WH_Act.Value |
| 3 - 4 | Not used |
| 5 | PV_WL_Act.Value |
| 6 | PV_AL_Act.Value |
| 7 | PV_AH_En OR Feature.Bit23 |
| 8 | PV_WH_En OR Feature.Bit23 |
| 9 - 10 | Not used |
| 11 | PV_WL_En OR Feature.Bit23 |
| 12 | PV_AL_En OR Feature.Bit23 |
| 13 | PV_AH_MsgEn |
| 14 | PV_WH_MsgEn |
| 15 - 16 | Not used |
| 17 | PV_WL_MsgEn |
| 18 | PV_AL_MsgEn |
| 19 - 28 | Not used |
| 29 | 1= Suppressed messages |
| 30 | Internal error |
| 31 | MS_RelOp |

### See also

MonAnS operating modes (Page 752)
MonAnS functions (Page 753)
MonAnS error handling (Page 755)

MonAnS messaging (Page 756)

MonAnS block diagram (Page 762)

## 5.4.2 MonAnS operating modes

### MonAnS operating modes

The block can be operated using the following modes:

- On (Page 77)
- Out of service (Page 77)

### "On"

General information on the "On" mode is available in the section On (Page 77).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit5` =0 and `Feature.Bit5` =1 is as shown below:

| Output Parameters | **`Feature.Bit5` =0** | **`Feature.Bit5` =1** |
|---|---|---|
| PV_Out.ST | 16#80 | 16#60 |
| PV_AH_Act.ST | 16#XX | 16#60 |
| PV_WH_Act.ST | 16#XX | 16#60 |
| PV_WL_Act.ST | 16#XX | 16#60 |
| PV_AL_Act.ST | 16#XX | 16#60 |

### See also

Description of MonAnS (Page 750)

MonAnS functions (Page 753)

MonAnS error handling (Page 755)

MonAnS messaging (Page 756)

MonAnS I/Os (Page 758)

MonAnS block diagram (Page 762)

## 5.4.3 MonAnS functions

### Functions of MonAnS

The functions for this block are listed below.

### Alarm delays with one time value per limit pair

This block includes the standard function alarm delay for One time value per limit pair (Page 226).

### Limit monitoring of the process value

This block provides the standard function Limit monitoring of the process value (Page 93).

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Group display SumMsgAct for limit monitoring, CSF and ExtMsgx

The block has the standard function Group display for limit monitoring, CSF and ExtMsgx (Page 92).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `PV_Out.ST`

### Dead band

To suppress values that fluctuate around the zero point, this block has the standard function Dead band (Page 67).

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 3 | Disable limit act output when limit value message is disabled (Page 209) |
| 5 | Changes signal status of outputs in OOS (Page 208) |
| 23 | Limits disabled in faceplate (Page 206) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 - 9 | Not used |
| 10 | 1 = Operator can change the simulation value `SimPV` |
| 11 | 1 = Operator can activate the Simulation function |
| 12 | 1 = Operator can activate the Release for maintenance function |
| 13 | 1 = Operator can change the limit (`PV`) for high alarm |
| 14 | 1 = Operator can change the limit (`PV`) for high warning |
| 15 | Not used |
| 16 | 1 = Operator can change the limit (`PV`) for hysteresis |
| 17 | 1 = Operator can change the limit (`PV`) for low alarm |
| 18 | 1 = Operator can change the limit (`PV`) for low warning |
| 19 - 22 | Not used |
| 23 | 1 = Operator can change the dead band parameter `DeadBand` |
| 24 | 1 = Operator can activate / deactivate messages via `PV_AH_MsgEn` |
| 25 | 1 = Operator can activate / deactivate messages via `PV_AL_MsgEn` |
| 26 -27 | Not used |

| Bit | Function |
|---|---|
| 28 | 1 = Operator can activate / deactivate messages via PV_WH_MsgEn |
| 29 | 1 = Operator can activate / deactivate messages via PV_WL_MsgEn |
| 30 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in OS_Perm as a parameter, you have to reset the corresponding OS_Perm bit.

### Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 230).

### Specifying the display area for process and setpoint values as well as operations

This block provides the standard function Display and operator input area for process values and setpoints (Page 233).

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of MonAnS (Page 750)

MonAnS operating modes (Page 752)

MonAnS error handling (Page 755)

MonAnS messaging (Page 756)

MonAnS I/Os (Page 758)

MonAnS block diagram (Page 762)

## 5.4.4 MonAnS error handling

### Error handling of MonAnS

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following error message can be generated at this block:

- Error numbers
- Control system fault (CSF)

### Overview of error numbers

The ErrorNum I/O can be used to output various error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed. |
| 0 | There is no error. |
| 30 | The value of PV can no longer be displayed in the REAL number field. |

### Control system fault (CSF)

An external signal can be activated via the CSF input. If this signal changes to = 1, a control system fault is triggered. Refer to the Error handling (Page 128) section for more on this.

### See also

Description of MonAnS (Page 750)

MonAnS operating modes (Page 752)

MonAnS functions (Page 753)

MonAnS messaging (Page 756)

MonAnS I/Os (Page 758)

MonAnS block diagram (Page 762)

## 5.4.5 MonAnS messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Process messages
- Instance-specific messages

### Process control fault

You can interconnect an external fault (signal) to input parameter CSF. If it changes to CSF = 1, a process control fault is triggered (MsgEvId1, SIG 5).

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 1 | Alarm - high | $$BlockComment$$<br>PV - high alarm limit violated |
| | SIG 2 | Warning - high | $$BlockComment$$<br>PV - high warning limit violated |
| | SIG 3 | Warning - low | $$BlockComment$$<br>PV - low warning limit violated |
| | SIG 4 | Alarm - low | $$BlockComment$$<br>PV - low alarm limit violated |
| | SIG 5 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Instance-specific messages

You have the option to use two instance-specific messages for this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 6 | AS process control message - fault | $$BlockComment$$<br>External message 1 |
| | SIG 7 | AS process control message - fault | $$BlockComment$$<br>External message 2 |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance **MsgEvId1**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |
| 4 | ExtVa104 |
| 5 | ExtVa105 |
| 6 | Reserved |
| 7 | Reserved |
| 8 | Reserved |
| 9 | Reserved |
| 10 | Reserved |

The associated values 4 ... 5 are allocated to the parameters `ExtVa104` ... `ExtVa105` and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of MonAnS (Page 750)
MonAnS operating modes (Page 752)
MonAnS functions (Page 753)
MonAnS error handling (Page 755)
MonAnS I/Os (Page 758)
MonAnS block diagram (Page 762)

## 5.4.6 MonAnS I/Os

MonAnS I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `BatchEn` | 1 = Enable allocation | BOOL | 0 |
| `BatchID` | Batch ID | DWORD | 16#00000000 |
| `BatchName` | Batch name | S7-String | |
| `BlkTypeInfo*` | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| `CSF` | 1 = External error (control system fault)<br>Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `DeadBand` | Width of dead band | REAL | 0.0 |
| `EN` | 1 = Called block will be processed | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa104 | Associated value 4 for messages (MsgEvID1) | ANY | |
| ExtVa105 | Associated value 5 for messages (MsgEvID1) | ANY | |
| Feature | I/O for additional MonAnS functions (Page 753) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| MS_RelOp* | 1 = Release for maintenance by OS operator | BOOL | 0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, Description of OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for MonAnS functions (Page 753) | STRUCT<br>• Bit 0: BOOL<br>• Bit 10: BOOL<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| PV* | Process value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_A_DC* | Delay time for incoming PV alarms [s] | REAL | 0.0 |
| PV_AH_En | 1 = Enable PV alarm limit (high) | BOOL | 1 |
| PV_AH_Lim | Limit PV alarm (high) | REAL | 95.0 |
| PV_AH_MsgEn | 1 = Enable PV alarm (high) message | BOOL | 1 |
| PV_AL_En | 1 = Enable PV alarm limit (low) | BOOL | 1 |
| PV_AL_Lim | PV alarm limit (low) | REAL | 5.0 |
| PV_AL_MsgEn | 1 = Enable PV alarm (low) message | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_Hyst* | Hysteresis for PV alarm, warning and tolerance limits | REAL | 0.0 |
| PV_OpScale | Limit for scale in PV bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| PV_Unit | Unit of measure for process value | INT | 1001 |
| PV_W_DC* | Delay time for incoming PV warnings [s] | REAL | 0.0 |
| PV_WH_En | 1 = Enable PV warning limit (high) | BOOL | 1 |
| PV_WH_Lim | Limit PV warning (high) | REAL | 90.0 |
| PV_WH_MsgEn | 1 = Enable PV warning (high) message | BOOL | 1 |
| PV_WL_En | 1 = Enable PV warning limit (low) | BOOL | 1 |
| PV_WL_Lim | Limit PV warning (low) | REAL | 10.0 |
| PV_WL_MsgEn | 1 = Enable PV warning (low) message | BOOL | 1 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | 1 = Call a block saved in this parameter as an Opening additional faceplates (Page 234) in the standard view | ANY | - |
| SimOn | 1 = Simulation on | BOOL | 0 |
| SimPV* | Process value used for SimOn = 1 | REAL | 0.0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see MonAnS error handling (Page 755) | INT | -1 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feed-forwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| PV_AH_Act | 1 = PV alarm (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_AL_Act | 1 = PV alarm (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Out | Output for process value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_OpScaleOut | PV_OpScale as output | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| PV_UnitOut | PV_Unit as output | INT | 0 |
| PV_WH_Act | 1 = PV warning (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_WL_Act | 1 = PV warning (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Description of MonAnS (Page 750) | DWORD | 16#00000000 |
| Status2 | Description of MonAnS (Page 750) | DWORD | 16#00000000 |
| SumMsgAct | 1 = Group message is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

**See also**

MonAnS operating modes (Page 752)
MonAnS messaging (Page 756)
MonAnS block diagram (Page 762)

## 5.4.7 MonAnS block diagram

### MonAnS block diagram

A block diagram is not provided for this block.

**See also**

Description of MonAnS (Page 750)
MonAnS operating modes (Page 752)
MonAnS functions (Page 753)
MonAnS error handling (Page 755)
MonAnS messaging (Page 756)
MonAnS I/Os (Page 758)

## 5.4.8 Operator control and monitoring

### 5.4.8.1 MonAnS views

### Views of the MonAnS block

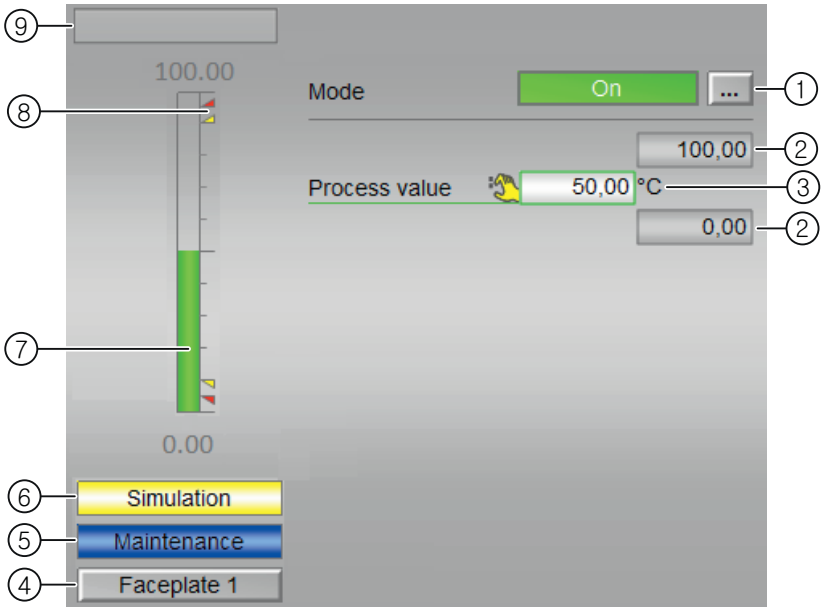
The block MonAnS provides the following views:

- MonAnS standard view (Page 763)
- Alarm view (Page 336)
- MonAnS limit view (Page 765)
- Trend view (Page 341)
- MonAnS parameter view (Page 766)
- MonAnS preview (Page 767)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for MonAnS (Page 768)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 5.4.8.2 MonAnS standard view

#### MonAnS standard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 77)
- Out of service (Page 77)

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the operating mode.

#### (2) High and low scale range for the process value

These values provide information on the display range for the bar graph of the process value. The scale range is defined in the engineering system.

#### (3) Display of the process value including signal status

This area shows the current process value with the corresponding signal status.

If text is configured for this command, it is displayed as additional text and button label for command selection. You can find more information about this in the section Labeling of buttons and text (Page 235).

You can change the text for the process value with the `PV_Out` parameter.

### (4) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

Additional information is available in the section Opening additional faceplates (Page 234).

### (5) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

### (6) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"

### (7) Bar graph for the "process value"

This area shows the current "Process value" in the form of a bar graph. You can also see the high and low limits of the bargraph's scale.The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (8) Limit display

These colored triangles show you the configured limits in the respective bar graph.

### (9) Display area for block states

This area provides additional information on the operating state of the block:

- Internal error
- "Invalid Signal"

### See also

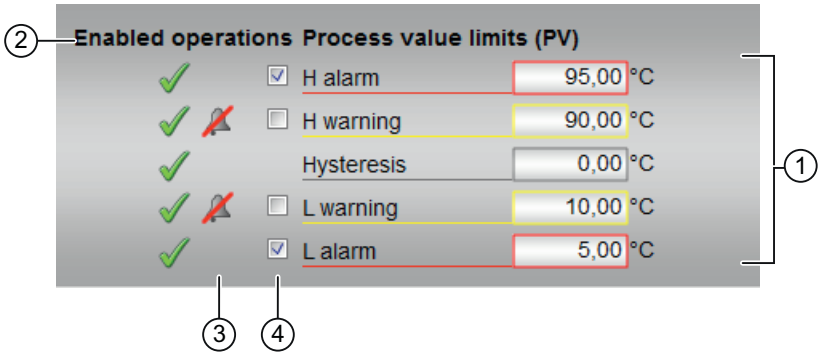
Displaying auxiliary values (Page 239)

### 5.4.8.3 MonAnS limit view

#### Limit view of MonAnS

You can specify the process value limits in this view:

The toolbars of the faceplate and the block icon indicate when the limits are reached or violated.




#### (1) Process value limits

In this area, you can enter the limits for the process value. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H alarm": Alarm high
- "H warning": Warning high
- "Hysteresis"
- "L warning": Warning low
- "L alarm": Alarm low

#### (2) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

### (3) Message suppression / delay

Message suppression indicates whether or not the suppression of the associated message in the AS block is activated with the `xx_MsgEn` parameter. The output of messages is not suppressed when the block is installed (all `xx_MsgEn` parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

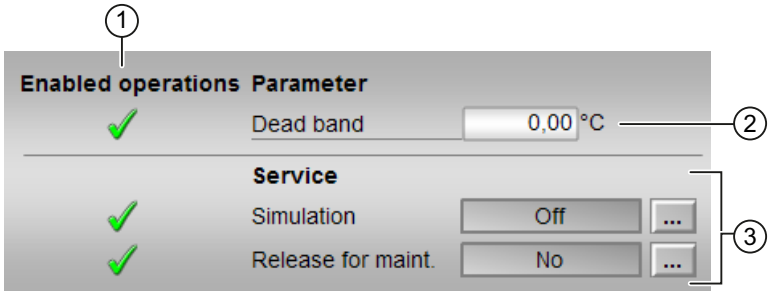
Alarm delays are also displayed in this position; for more on this see section Area of application for alarm delays (Page 225).

### (4) Suppress messages

You can enable / disable messages by setting the check mark.

## 5.4.8.4 MonAnS parameter view

### Parameter view of MonAnS




### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

### (2)Parameter

You can change parameters in this area. Refer to the section Changing values (Page 291).

You can influence the following parameters:

- "Dead band": Width of dead band

### (3)Service

You can select the following functions in this area:

- "Simulation"
- "Release for maintenance"

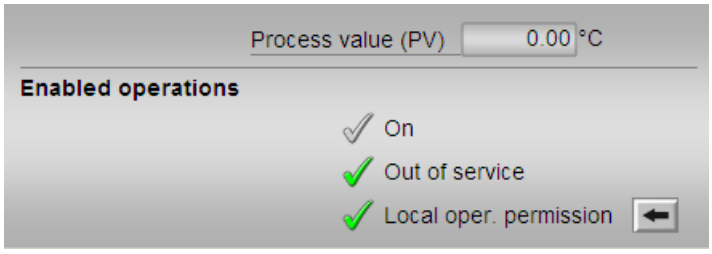
Refer to the Switching operating states and operating modes (Page 289) section for more on this.

You can find information on this area in the section:

- Simulating signals (Page 64)
- Release for maintenance (Page 70)

## 5.4.8.5 MonAnS preview

### Preview of MonAnS




### Process value

This area displays the real process value (`PV`).

### Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. Additional information is available in the section Operator control permissions (Page 285).

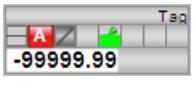
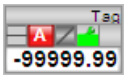
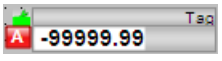
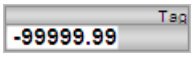
### 5.4.8.6 Block icon for MonAnS

#### Block symbols for MonAnS

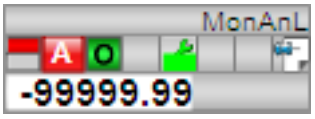
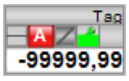
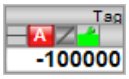
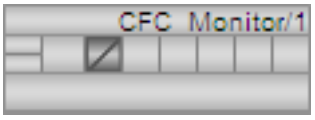
A variety of block symbols are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as control system faults
- Operating modes
- Signal status, release for maintenance
- Memo display
- Process value

The block symbols from template @TemplateAPLV8.PDL:

| Symbols | Selection of the block symbol in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | |
|  | 4 | |

The block symbols from template @TemplateAPLV7.PDL:

| Symbols | Selection of the block symbol in CFC | Special features |
|---|---|---|
|  | 1 | Block symbol in the full display |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | - | Block symbol in "Out of service" mode (example with type 1 block symbol) |

Additional information on the block symbol and the control options in the block symbol is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269).

# 5.5 MonDiL - Monitoring of a digital process tag (Large)

## 5.5.1 Description of MonDiL

### Object name (type + number) and family

Type + number: FB 1848

Family: Monitor

### Area of application for MonDiL

The block is used for the following applications:

- Monitoring a digital process tag

**Note**

This block is also available as a small block. A comparison of the MonDiL and MonDiS blocks is available in the section: MonDiL compared to MonDiS (Page 703)

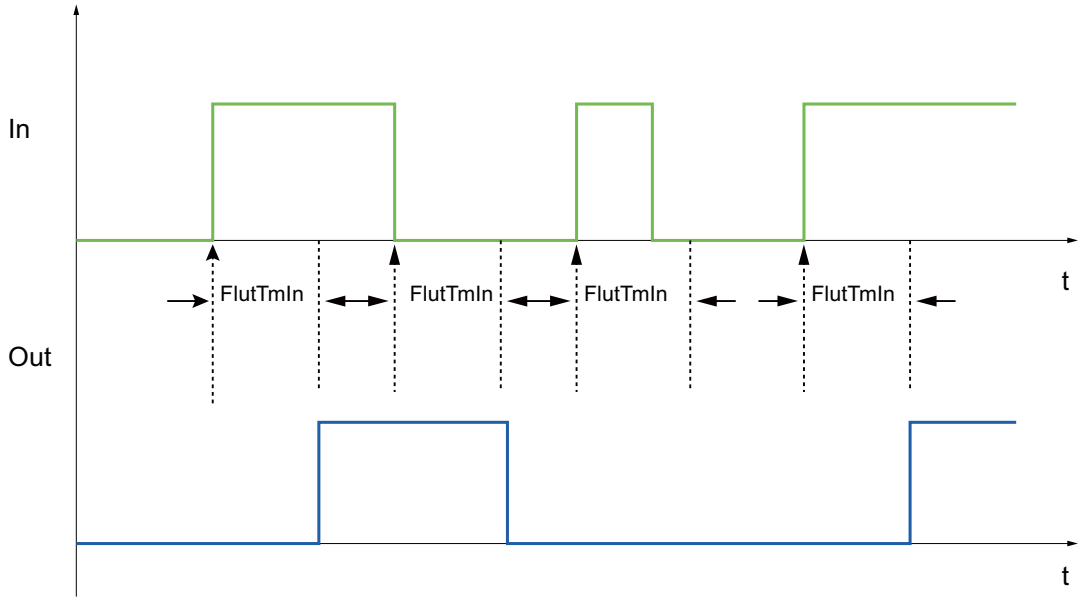
### How it works

The MonDiL block is used to monitor a digital process tag with flutter suppression. The block reports excess flutter signals which are generated within a defined period.

The digital value to be monitored is interconnected to the `In` input parameter. Every time there is a signal change (1 - 0 or 0 - 1) the configurable timer (`FlutTmIn`) for flutter suppression is started as shown in the figure below.

When the time you have specified expires and no single change occurs, the input signal is written to the `Out` output parameter.

Set the time in the timer (`FlutTmIn`) to 0 seconds; the input signal is written directly to the output.




## Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

For the MonDiL block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Monitoring of a digital process tag (DigitalMonitoring) (Page 2851)
- Monitoring a digital process tag for PA/FF devices (DigitalMonitoring_Fb) (Page 2852)

## Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block. The `Out` and `FlutAct` parameters are affected.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

## Status word allocation for `Status1` parameter

You can find a description for each parameter in section MonDiL I/Os (Page 781).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `BypassAct.Value` |

| Status bit | Parameter |
|---|---|
| 6 | `OnAct.Value` |
| 7 | `Out.Value` |
| 8 | `AlmMsgEn` |
| 9 | Display of `BypassAct.Value` in faceplate (display and operator controls) and block icon |
| 10 | `SimLiOp.Value` |
| 11 | Delay of Output set message |
| 12 - 29 | Not used |
| 30 | Auxiliary value 1 is visible |
| 31 | Auxiliary value 2 is visible |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `MsgLock.Value` |
| 1 | `FlutAct.Value` |
| 2 | `FlutEn` |
| 3 | `FlutMsgEn` |
| 4 - 8 | Not used |
| 9 | `InvIn` |
| 10 - 28 | Not used |
| 29 | 1= Suppressed messages |
| 30 | Internal error |
| 31 | `MS_RelOp` |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 | Effective signal 1 of the message block connected via `EventTsIn` |
| 1 | Effective signal 2 of the message block connected via `EventTsIn` |
| 2 | Effective signal 3 of the message block connected via `EventTsIn` |
| 3 | Effective signal 4 of the message block connected via `EventTsIn` |
| 4 | Effective signal 5 of the message block connected via `EventTsIn` |
| 5 | Effective signal 6 of the message block connected via `EventTsIn` |
| 6 | Effective signal 7 of the message block connected via `EventTsIn` |
| 7 | Effective signal 8 of the message block connected via `EventTsIn` |
| 8 - 13 | Not used |
| 14 | Color = 11 (Background color is derived directly from `EventState`, without message classes process message and operator prompt), only for block icon type 5 |
| 15 | Color = 10 (Background color of the message instance MsgEvId1 - Message identifier SIG1) |
| 16-19 | Not used |
| 20 | Color = 6 (Violet, operator prompt) and `Out = 1` |

| Status bit | Parameter |
|---|---|
| 21 | Not used |
| 22 | Color = 5 (Pastel green, process) and `Out = 1` |
| 23 - 24 | Not used |
| 25 | Color = 4 (Black, control system) and `Out = 1` |
| 26 | Color = 9 (UCMC 6) and `Out = 1` |
| 27 | Color = 3 (Blue, tolerance / UCMC 5) and `Out = 1` |
| 28 | Color = 8 (UCMC 4) and `Out = 1` |
| 29 | Color = 2 (Yellow, warning UCMC 3) and `Out = 1` |
| 30 | Color = 7 (UCMC 2) and `Out = 1` |
| 31 | Color = 1 (Red, alarm / UCMC 1) and `Out = 1` |

Explanation:
UCMC = user-configured messages classes

**Note**

The user-configured message classes 5, 3 and 1 are displayed in the existing `Status.Bit27`, `Status.Bit29` and `Status.Bit31`.

### Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 - 15 | Effective signal 9..16 of the message block connected via `EventTsIn` |
| 16 - 31 | Not used |

### See also

MonDiL functions (Page 774)
MonDiL messaging (Page 780)
MonDiL block diagram (Page 786)
MonDiL error handling (Page 779)
MonDiL modes (Page 773)

## 5.5.2 MonDiL modes

### MonDiL operating modes

This block provides the following modes.

- On (Page 77)
- Out of service (Page 77)

### "On"

You can find general information about the "On" mode in the On (Page 77) section.

### Out of service

You can find general information about the "Out of service" mode in the Out of service (Page 77) section.

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit5` =1 and `Feature.Bit5` =0 is as shown below:

| Output Parameters | **Feature.Bit5 =0** | **Feature.Bit5 =1** |
|---|---|---|
| Out.ST | 16#80 | 16#60 |

### See also

MonDiL block diagram (Page 786)

MonDiL I/Os (Page 781)

MonDiL messaging (Page 780)

MonDiL error handling (Page 779)

MonDiL functions (Page 774)

Description of MonDiL (Page 770)

## 5.5.3 MonDiL functions

### Functions of MonDiL

The functions for this block are listed below.

### Suppression and reporting of signal flutter

The block is operated as "flutter filter". The block receives digital signals at the `In` input parameter and ideally these do not develop any flutter. The block monitors this if you activate the function via the `FlutEn` = 1 input parameter.

Use the `FlutTmIn` input parameter to set how long a continuous signal should last in order to be transferred to the process without flutter.

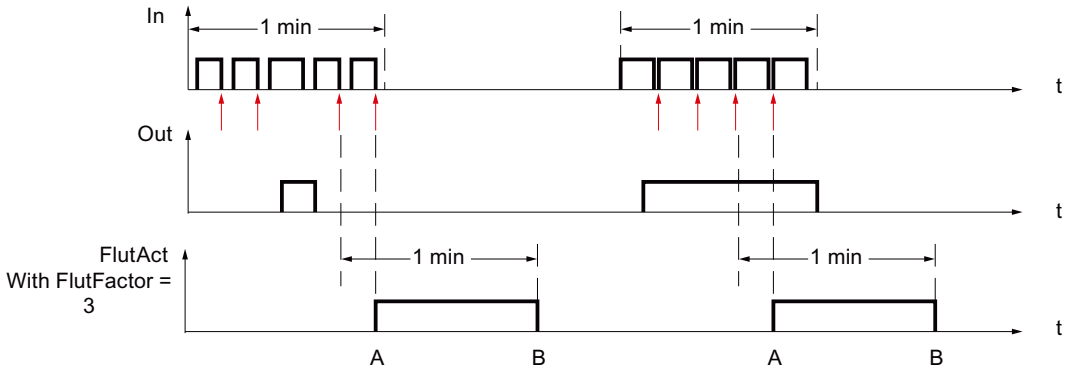
The preprocessed signal is sent to the process via the `Out` output parameter.

The block can be set up to report signal flutter. To do this, set parameter `FlutMsgEn` = 1. Monitoring starts at the next 0 - 1 - 0 edge transition at the input signal `In`.

Use the `FlutFactor` input parameter to specify the maximum number of signals to be filtered per minute by the block. If this maximum value is exceeded, this is indicated at the output parameter `FlutAct` by a 1. The message (`FlutAct` = 0). is cleared after the maximum value has dropped again by more than half.

### Example of the flutter suppression

You can use the `FlutFactor` input parameter to limit flutter signals to 3 per minute.




Case A:

Four flutter signals have occurred within a minute. `FlutFactor` is set to three, in other words, a maximum of three flutter signal per minute are allowed, the `FlutAct` output is set to one.

Case B:

Only one flutter signal has occurred within a minute, which corresponds to less than half the `FlutFactor`. The `FlutAct` output is reset.

### Delaying the on and off switching functions

You set delay times for setting/resetting the output using the input parameter `Out_DC` or `Out_DG`:

- `Out_DC`: delay time [s] for rising edges (0 - 1 edge)
- `Out_DG`: delay time [s] for falling edge (1 - 0 edge)

The `Out` output parameter is set/reset after expiration of the delay time.

You disable this function if you set the value of the respective parameter to 0 seconds.

### Adapting the color representation in the configured message class

You can set the background color of instance-specific texts for the block icons 2 to 5 and thereby adapt them to the configured message class. This background color is displayed when the output parameter `Out` = 1.

You set the color coding with the `Color` parameter:

| Value `Color` | Color |
|---|---|
| 0 | Old reaction |
| 1 | Red (alarm) |
| 2 | Yellow (warning) |
| 3 | Blue (tolerance) |
| 4 | Black (control system) |
| 5 | Pastel green (process) |

| Value `Color` | Color |
|---|---|
| 6 | Violet (operator prompt) |
| 7...9 | Old reaction |
| 10 | Background color of the message instance MsgEvId1 - Message identifier SIG1 |
| 11 | For block icon type 5 only, background color is derived directly from EventState without message classes process message and operator prompt [1] |

[1] This also takes into consideration the CSF input

The following colors are shown for user-configured message classes

| Value `Color` | Color |
|---|---|
| 0 | Old reaction |
| 1 | User-configured message classes 1 |
| 2 | User-configured message classes 3 |
| 3 | User-configured message classes 5 |
| 4 | Black (control system) |
| 5 | Pastel green (process) |
| 6 | Violet (operator prompt) |
| 7 | User-configured message classes 2 |
| 8 | User-configured message classes 4 |
| 9 | User-configured message classes 6 |
| 10 | Background color of the message instance MsgEvId1 - Message identifier SIG1 |
| 11 | For block icon type 5 only, background color is derived directly from EventState without message classes process message and operator prompt [1] |

[1] This also takes into consideration the CSF input

### Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 239).

### Changing labels on buttons and text

This block provides the standard function Labeling of buttons and text (Page 235).

You can change the label for "Process value" for this block as you please. The change is made with the `FlutTmIn` parameter.

### Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 230).

### Group display `SumMsgAct` for limit monitoring, `CSF` and `ExtMsgx`

The block provides the standard function Group display for limit monitoring, CSF and ExtMsgx (Page 92).

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `In.ST`

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Value (`SimIn`, `SimInLi`)

### Bypass function

This block provides the standard function Bypassing signals (Page 115).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 5 | Changes signal status of outputs in OOS (Page 208) |
| 9 | Substitution value is active if the block is in bypass (Page 199) |
| 11 | Suppress MsgLock and “Out of service” mode for a connected message block (Page 202) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can change the delay time for rising edges (`Out_DC`) |
| 5 | 1 = Operator can change the delay time for falling edges (`Out_DG`) |
| 6 - 7 | Not used |
| 8 | 1 = Operator can activate bypass functionality |
| 9 | 1 = Operator can deactivate bypass functionality |
| 10 | 1 =Operator can change the simulation value `SimIn` |
| 11 | 1 = Operator can activate the Simulation function |
| 12 | 1 = Operator can activate the Release for maintenance function |
| 13 | Not used |
| 14 | 1 = The user can input the duration for flutter suppression. |
| 15 | 1 = The user can specify the number of flutter signals that should be suppressed. |
| 16 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of MonDiL (Page 770)

MonDiL messaging (Page 780)

MonDiL I/Os (Page 781)

MonDiL block diagram (Page 786)

MonDiL error handling (Page 779)

MonDiL modes (Page 773)

EventTs functions (Page 2046)

Reaction of the switching points in the "Out of service" operating mode (Page 188)

User-configured message classes (Page 45)

## 5.5.4 MonDiL error handling

### Error handling of MonDiL

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Control system fault (CSF)
- Flutter alarm

### Overview of error numbers

The ErrorNum I/O can be used to output various error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed. |
| 0 | There is no error. |
| 11 | FlutTmIn < 0 |
| 16 | FlutFactor is -ve or > 100 |

### Control system fault (CSF)

An external signal can be activated via the CSF input. If this signal changes to = 1, a control system fault is triggered. Refer to the Error handling (Page 128) section for more on this.

### Flutter alarm

An alarm is output at output FlutAct with 1 if signal flutter is detected. Refer to the functions of the block > Error handling (Page 128).

### See also

MonDiL block diagram (Page 786)

MonDiL I/Os (Page 781)

MonDiL messaging (Page 780)

MonDiL functions (Page 774)
MonDiL modes (Page 773)
Description of MonDiL (Page 770)

## 5.5.5 MonDiL messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Process messages
- Instance-specific messages

### Process control fault

You can interconnect an external fault (signal) to input parameter `CSF`. If it changes to `CSF` = 1, a process control fault is triggered (`MsgEvId1`, SIG 3).

### Process messages

You can use up to four instance-specific messages with this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId1` | SIG 1 | Alarm - high | $$BlockComment$$<br>`Out` - Binary value set |
| | SIG 2 | Warning - high | $$BlockComment$$<br>Flutter limits violated |
| | SIG 3 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |
| | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External message 1 |
| | SIG 5 | AS process control message - fault | $$BlockComment$$<br>External message 2 |
| | SIG 6 | AS process control message - fault | $$BlockComment$$<br>External message 3 |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance `MsgEvId1`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | `ExtVa104` |
| 5 | `ExtVa105` |
| 6 | `ExtVa106` |
| 7 | `ExtVa107` |
| 8 | `ExtVa108` |
| 9 | Reserved |
| 10 | Reserved |

The associated values 4 ... 8 are allocated to the parameters `ExtVa104` ... `ExtVa108` and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of MonDiL (Page 770)

MonDiL functions (Page 774)

MonDiL I/Os (Page 781)

MonDiL block diagram (Page 786)

MonDiL error handling (Page 779)

MonDiL modes (Page 773)

Time stamp (Page 231)

## 5.5.6 MonDiL I/Os

I/Os of MonDiL

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `AlmMsgEn` | 1 = Alarms are output | BOOL | 1 |
| `BatchEn` | 1 = Enable allocation | BOOL | 0 |
| `BatchID` | Batch ID | DWORD | 16#00000000 |
| `BatchName` | Batch name | S7-String | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkTypeData* | This input can not be edit or linked and is for internal usage only! | | |
| BypLiOp | 1 = Bypass commands via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypLock | 1 = Bypass activation or deactivation is locked for operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypIn | Substitution value if block is in bypass | REAL | 0.0 |
| BypInLi | 1 = Select bypass In (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypInOp | 1 = Select bypass In (via operator) | BOOL | 0 |
| CSF | 1 = External error (control system fault)<br>Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Color | Display for status display in the block icon:<br>0 = Default setting<br>1 = Red (alarm)<br>2 = Yellow (warning)<br>3 = Blue (tolerance)<br>4 = Black (control system message)<br>5 = Pastel green (process message)<br>6 = Violet (operator prompt) | BYTE | 16#00 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| EventTsIn | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg3 | Binary input for freely selectable message 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa104 | Associated value 4 for messages (MsgEvID1) | ANY | |
| ExtVa105 | Associated value 5 for messages (MsgEvID1) | ANY | |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| ExtVa108 | Associated value 8 for messages (MsgEvID1) | ANY | |
| Feature | I/O for additional functions (Page 774) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| FlutEn | 1 = Flutter suppression on | BOOL | 1 |
| FlutFactor | Number of flutter signals that can be suppressed | INT | 2 |
| FlutMsgEn | 1 = Signal flutter is reported if the period of the signal flutter is < FlutTmIn. | BOOL | 1 |
| FlutTmIn | Period during which signal flutter is suppressed. | REAL | 0.0 |
| In | Digital input value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| InvIn | Invert input In | BOOL | 0 |
| MS_RelOp* | 1 = Release for maintenance by OS operator | BOOL | 0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OS_Perm | I/O for operator permissions (Page 774) | STRUCT<br>• Bit 0: BOOL<br>• Bit 10: BOOL<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| Out_DC | Delay time for setting the output parameter Out for rising edges | REAL | 0.0 |
| Out_DG | Delay time for resetting the output parameter Out for falling edges | REAL | 0.0 |
| RstBypLi | 1 = Reset bypass In (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstBypOp | 1 = Reset bypass In (via operator) | BOOL | 0 |
| RunUpCyc | Messages can be suppressed for a short time after start-up. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | 1 = Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SelFp2 | 1 = Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | - |
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SimIn* | Value used for SimOn = 1 | BOOL | 0 |
| SimInLi | Value that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserAna1 | Analog auxiliary value 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UA1unit | Unit of measure for analog auxiliary value 1 | INT | 0 |
| UserAna2 | Analog auxiliary value 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UA2unit | Unit of measure for analog auxiliary value 2 | INT | 0 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| BypassAct | 1 = Bypass is activated in this block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see MonDiL error handling (Page 779) | INT | -1 |
| FlutAct | 1 = Suppresses flutter signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message acknowledgement status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | 1 = Alarm error 1 (output ERROR of the first ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feed-forwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Out | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 770) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 770) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 770) | DWORD | 16#00000000 |
| SumMsgAct | 1 = Group message is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| UA1unitOut | UA1 as output | INT | 0 |
| UA2unitOut | UA2 as output | INT | 0 |
| UserAna1Out | UserAna1 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| UserAna2Out | UserAna2 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |

### See also

MonDiL messaging (Page 780)

MonDiL block diagram (Page 786)

MonDiL modes (Page 773)

## 5.5.7 MonDiL block diagram

### MonDiL block diagram

A block diagram is not provided for this block.

**See also**

MonDiL I/Os (Page 781)
MonDiL messaging (Page 780)
MonDiL error handling (Page 779)
MonDiL functions (Page 774)
MonDiL modes (Page 773)
Description of MonDiL (Page 770)

## 5.5.8 Operator control and monitoring

### 5.5.8.1 MonDiL views

**Views of the MonDiL block**

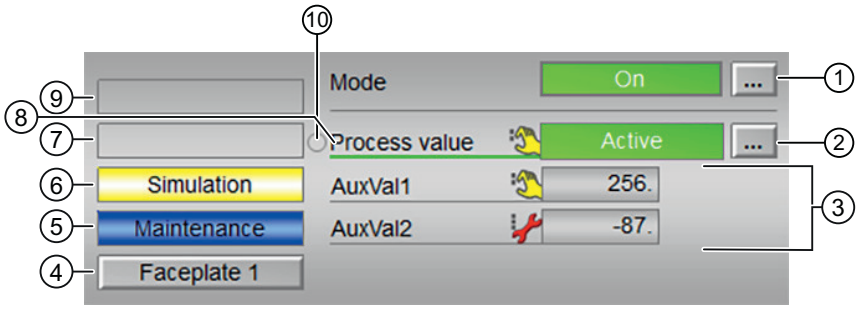
The block MonDiL provides the following views:

- MonDiL standard view (Page 788)
- Alarm view (Page 336)
- Trend view (Page 341)
- MonDiL parameter view (Page 790)
- MonDiL preview (Page 792)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for MonDiL (Page 793)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 5.5.8.2 MonDiL standard view

#### MonDiL standard view




#### (1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 77)
- Out of service (Page 77)

Refer to chapter Switching operating states and operating modes (Page 289) for information on switching the operating mode.

#### (2) Display of process value enabled/disabled

This area shows you the status of the individual connected parameters.

The identifier can be changed using Text 0 / Text 1 at the output parameter `Out`.

If the block is in simulation, you can enable or disable the process value. To do this, click on the display to open the operator input area.

If text is configure for these commands, it is displayed as additional text and as button labels for command selection. You can find additional information on this in chapter Labeling of buttons and text (Page 235).

You can change the text for the process value with the `FlutTmIn` parameter.

The background color of the display can be changed using the "Color" parameter; see MonDiL functions (Page 774) under "Adapting the color representation in the configured message class".

#### (3) Display of auxiliary values

You can use this area to display two auxiliary values that have been configured in the Engineering System. You can find additional information on this in chapter Displaying auxiliary values (Page 239).

### (4) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the Engineering System (ES). The visibility of this navigation button depends on the configuration in the Engineering System (ES).

You can find additional information on this in chapter Opening additional faceplates (Page 234).

### (5) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

### (6) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"

### (7) Display area for block states

This area provides additional information on the operating state of the block:

- "Fluttering"

### (8) Displaying input values

**Changing the display:**

Follow these steps to change the displays:

1. In the CFC, right-click the MonDiL block and select "Object Properties" from the context menu.
2. Select "I/Os" tab in the properties window.
3. In the “OS additional text” column, change the default setting for the input parameter (`FlutTmIn`) to what you want to see during runtime later.
4. In the "Identifier" column, change the default setting for the input parameter (`FlutTmIn`) to what you want to see during runtime later.
5. If both “OS additional text” and “Identifier” columns are empty, the default value is displayed in the runtime.
6. The text is used as a label and is therefore always displayed, which means it is independent of the signal status of the corresponding input (`FlutTmInx`).
7. The font size is reduced during the runtime if the input text length is greater than the label width.

**Note**

If a text is added to the "OS additional text" field of the input parameter (`FlutTmIn`), this text is displayed even if the "Identifier" text field of the same input parameter is not empty. If the "OS additional text" field of the input parameter (`FlutTmIn`) is empty, the "Identifier" text of the same input parameter is displayed.

### (9) Display area for block states

This area provides additional information on the operating state of the block:

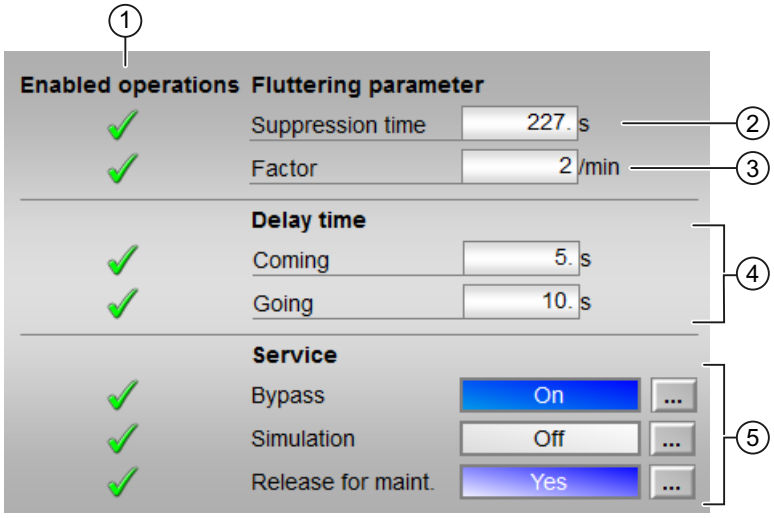
- Internal error

### (10) Display of Inverted Input

Blank circle will be displayed when input signal In is inverted with `InvIn`.

## 5.5.8.3 MonDiL parameter view

### Parameter view of MonDiL




### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

### (2) Suppression time

Enter the time period during which signal flutter is suppressed. You can find additional information on this in the Changing values (Page 291) section.

### (3) Factor

Enter the number of flutter signals that can be suppressed. You can find additional information on this in the Changing values (Page 291) section.

### (4) Delay time

Enter here the delay time by which the output should be set. Enter delay times here for positive ("incoming", 0 - 1 edge) and negative edges ("outgoing", 1 - 0 edge). You can find additional information on this in the Changing values (Page 291) section.

### (5) Service

You can select the following functions in this area:

- "Bypass"
- "Simulation"
- "Release for maintenance"

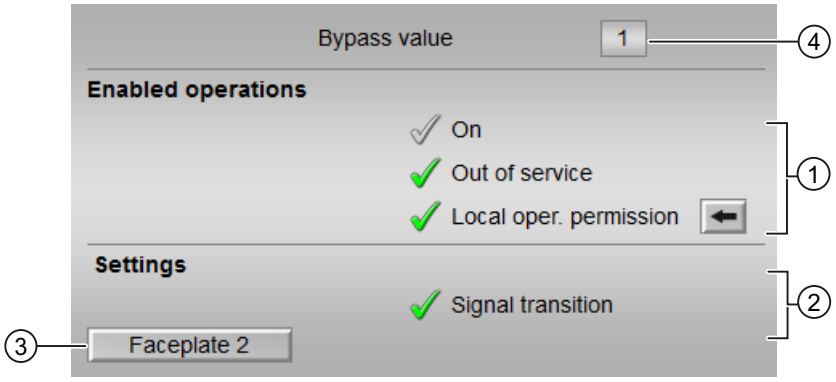
Refer to the Switching operating states and operating modes (Page 289) section for more on this.

You can find information on this area in the section:

- Bypassing signals (Page 115)
- Simulating signals (Page 64)
- Release for maintenance (Page 70)

### 5.5.8.4 MonDiL preview

#### Preview of MonDiL




#### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. Additional information is available in the section Operator control permissions (Page 285).

#### (2) Settings

- "Signal transition":
  - Activated: A message is generated with a "0 → 1" signal transition at the monitored input.
  - Deactivated: No message is generated.

#### (3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Opening additional faceplates (Page 234) section.

### (4) Bypass value

This area displays the bypass value (`BypIn`).

## 5.5.8.5 Block icon for MonDiL

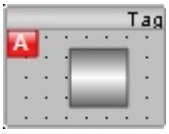
### Block icons for MonDiL

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as control system faults
- Operating modes
- Signal status, release for maintenance
- Memo display
- Display of the output signal
- Display configured instance-specific text for the process value (only for block icons 2 to 5). The instance-specific text for the process value can be changed using Text 0 / Text 1 at the output parameter `Out`.

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | The text background color for output `Out = 1` depends on the parameter assignment of the input "`Color`".<br>Text in block icon can be configured via Text 1 at the `Out` output parameter. |
|  | 3 | The text background color for output `Out = 1` depends on the parameter assignment of the input "`Color`".<br>Text in block icon can be configured via Text 1 at the `Out` output parameter. |

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 4 | The text background color for output `Out = 1` depends on the parameter assignment of the input "`Color`".<br>Text in block icon can be configured via Text 1 at the `Out` output parameter, a maximum of two characters are supported. |
|  | 5 | The text background color for output `Out = 1` depends on the parameter assignment of the input "`Color`".<br>Text in block icon can be configured via Text 1 at the `Out` output parameter. For `Color = 11`, the text background color is directly derived from EventState (but without the message classes process message and operator prompt) which also takes into consideration the CSF input. `Color = 11` is only supported by type 5. |
|  | 6 | With this block icon you can use your own e.g. emf-pictures. See "Customer-specific icons" in chapter Block icon structure (Page 260) |

**Note**

Text 1 is also used in the standard view of the faceplate in the display process value active/inactive. Long static texts must therefore be displayed on the "Process value" text using the "`FlutTmIn`" parameter (see MonDiL standard view (Page 788))

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 3 | The text background color for output `Out = 1` depends on the parameter assignment of the input "`Color`". |
|  | 4 | See 3 |
|  | 5 | See 3 |
|  | 6 | See 3 |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269).

## MonDiL6 block icon state display

The MonDiL6 block icon states are shown here:

| Symbol | Meaning |
|---|---|
|  | Active |
|  | Inactive |
|  | Out of service |

# 5.6 MonDiS - Monitoring of a digital process tag (Small)

## 5.6.1 Description of MonDiS

### Object name (type + number) and family

Type + number: FB 1913

Family: Monitor

### Area of application for MonDiS

The block is used for the following applications:

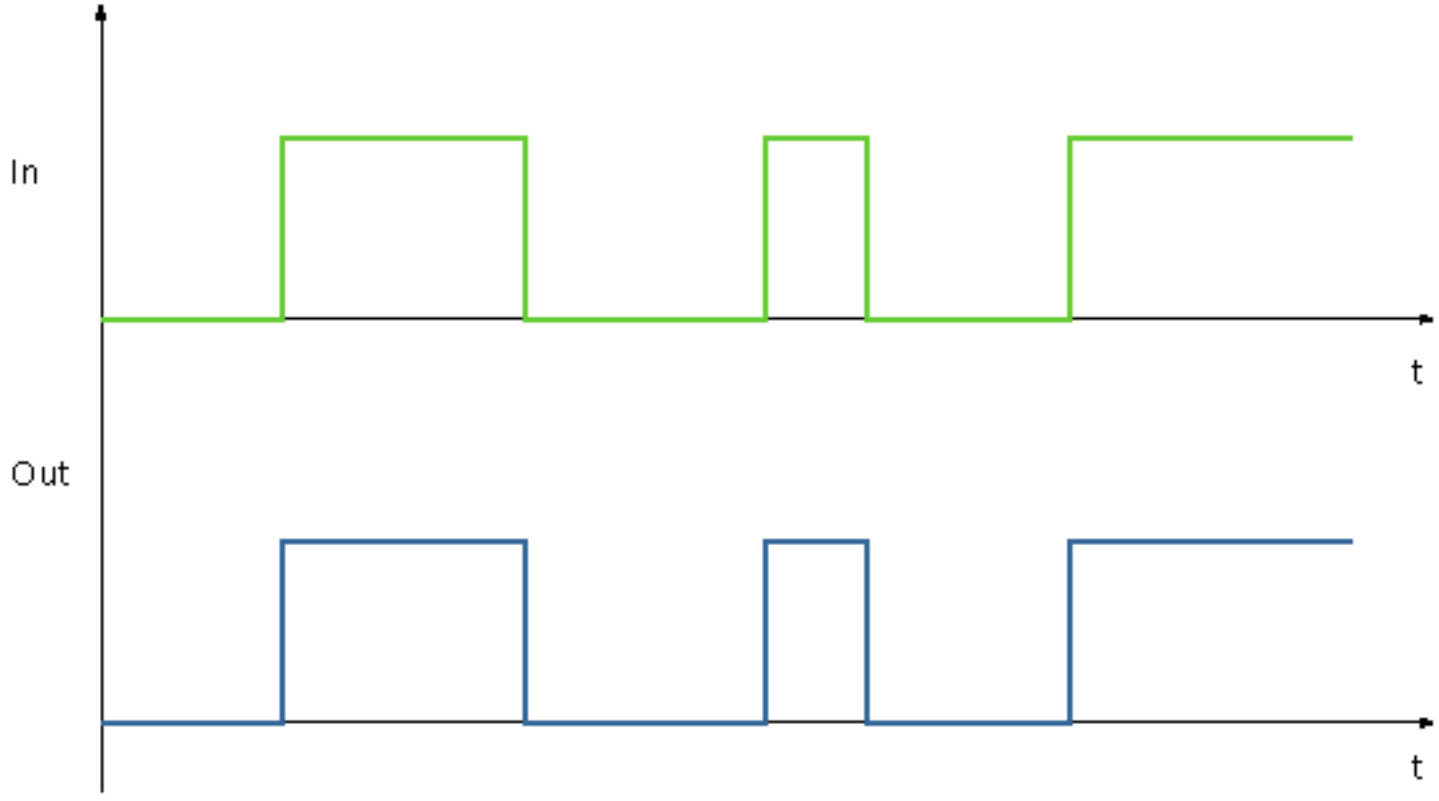
- Monitoring a digital process tag

**Note**

This block is also available as a large block. A comparison of the MonDiL and MonDiS blocks is available in the section: MonDiL compared to MonDiS (Page 703).

## How it works

The MonDiS is used to monitor a digital process tag. The digital value to be monitored is interconnected to the `In` input parameter. The input signal is written directly to the `Out` output parameter.




## Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

## Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block. The `Out` parameter is affected.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

## Status word allocation for `Status1` parameter

You can find a description for each parameter in section MonDiS I/Os (Page 805).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |

| Status bit | Parameter |
|---|---|
| 2 | SimOn |
| 3 | OosAct.Value |
| 4 | OosLi.Value |
| 5 | Not used |
| 6 | OnAct.Value |
| 7 | Out.Value |
| 8 | AlmMsgEn |
| 9 | Delay of Output set message |
| 10 - 31 | Not used |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | MsgLock.Value |
| 1 - 8 | Not used |
| 9 | InvIn |
| 10 - 28 | Not used |
| 29 | 1= Suppressed messages |
| 30 | Internal error |
| 31 | MS_RelOp |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 - 13 | Not used |
| 14 | Color = 11 (Background color is derived directly from EventState, without message classes process message and operator prompt), only for block icon type 5 |
| 15 | Color = 10 (Background color of the message instance MsgEvId1 - Message identifier SIG1) |
| 16-19 | Not used |
| 20 | Color = 6 (Violet, operator prompt) and Out = 1 |
| 21 | Not used |
| 22 | Color = 5 (Pastel green, process) and Out = 1 |
| 23 - 24 | Not used |
| 25 | Color = 4 (Black, control system) and Out = 1 |
| 26 | Color = 9 (UCMC 6) and Out = 1 |
| 27 | Color = 3 (Blue, tolerance / UCMC 5) and Out = 1 |
| 28 | Color = 8 (UCMC 4) and Out = 1 |
| 29 | Color = 2 (Yellow, warning / UCMC 3) and Out = 1 |

| Status bit | Parameter |
|---|---|
| 30 | Color = 7 (UCMC 2) and `Out = 1` |
| 31 | Color = 1 (Red, alarm / UCMC 1) and `Out = 1` |

Explanation:
UCMC = user-configured messages classes

**Note**

The user-configured message classes 5, 3 and 1 are displayed in the existing `Stauts.Bit27`, `Status.Bit29` and `Status.Bit31`.

### See also

MonDiS operating modes (Page 799)

MonDiS functions (Page 800)

MonDiS error handling (Page 803)

MonDiS messaging (Page 804)

MonDiS block diagram (Page 808)

## 5.6.2 MonDiS operating modes

### MonDiS operating modes

This block provides the following modes.

- On (Page 77)
- Out of service (Page 77)

### "On"

General information on the "On" mode is available in the section On (Page 77).

### Out of service

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit5` =1 and `Feature.Bit5` =0 is as shown below:

| Output Parameters | **`Feature.Bit5 =0`** | **`Feature.Bit5 =1`** |
|---|---|---|
| Out.ST | 16#80 | 16#60 |

### See also

Description of MonDiS (Page 796)
MonDiS functions (Page 800)
MonDiS error handling (Page 803)
MonDiS messaging (Page 804)
MonDiS I/Os (Page 805)
MonDiS block diagram (Page 808)

## 5.6.3 MonDiS functions

### Functions of MonDiS

The functions for this block are listed below.

### Delaying the on function

You set delay times for setting the output using the input parameter `Out_DC`:

- `Out_DC`: delay time [s] for rising edges (0 - 1 edge)

The `Out` output parameter is after expiration of the delay time.

You disable this function if you set the value of the respective parameter to 0 seconds.

### Adapting the color representation in the configured message class

You can set the background color of instance-specific texts for the block icons 2 to 5 and thereby adapt them to the configured message class. This background color is displayed when the output parameter `Out` = 1.

You set the color coding with the `Color` parameter:

| Value `Color` | Color |
|---|---|
| 0 | Old reaction |
| 1 | Red (alarm) |
| 2 | Yellow (warning) |
| 3 | Blue (tolerance) |
| 4 | Black (control system) |
| 5 | Pastel green (process) |
| 6 | Violet (operator prompt) |
| 7...9 | Old reaction |
| 10 | Background color of the configured message class SIG1 ("Out - setting binary output values") |
| 11 | For block icon type 5 only, background color is derived directly from EventState without message classes process message and operator prompt [1] |

[1] This also takes into consideration the CSF input

The following colors are shown for user-configured message classes

| Value `Color` | Color |
|---|---|
| 0 | Old reaction |
| 1 | User-configured message classes 1 |
| 2 | User-configured message classes 3 |
| 3 | User-configured message classes 5 |
| 4 | Black (control system) |
| 5 | Pastel green (process) |
| 6 | Violet (operator prompt) |
| 7 | User-configured message classes 2 |
| 8 | User-configured message classes 4 |
| 9 | User-configured message classes 6 |
| 10 | Background color of the configured message class SIG1 ("Out - setting binary output values") |
| 11 | For block icon type 5 only, background color is derived directly from EventState without message classes process message and operator prompt [1] |

[1] This also takes into consideration the CSF input

### Changing labels on buttons and text

This block provides the standard function Labeling of buttons and text (Page 235).

You can change the label for "Process value" for this block as you please. The change is made with the `OnOp` parameter.

### Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 230).

### Group display `SumMsgAct` for limit monitoring, `CSF` and `ExtMsgx`

The block provides the standard function Group display for limit monitoring, CSF and ExtMsgx (Page 92).

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `In.ST`

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 5 | Changes signal status of outputs in OOS (Page 208) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can change the delay time for rising edges (`Out_DC`) |
| 5 - 9 | Not used |
| 10 | 1 =Operator can change the simulation value `SimIn` |
| 11 | 1 = Operator can activate the Simulation function |
| 12 | 1 = Operator can activate the Release for maintenance function |
| 13 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of MonDiS (Page 796)

MonDiS operating modes (Page 799)

MonDiS error handling (Page 803)

MonDiS messaging (Page 804)

MonDiS I/Os (Page 805)

MonDiS block diagram (Page 808)

User-configured message classes (Page 45)

## 5.6.4 MonDiS error handling

### Error handling of MonDiS

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Control system fault (CSF)

### Overview of error numbers

The ErrorNum I/O can be used to output various error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed. |
| 0 | There is no error. |

### Control system fault (CSF)

An external signal can be activated via the CSF input. If this signal changes to = 1, a control system fault is triggered. Refer to the Error handling (Page 128) section for more on this.

### See also

Description of MonDiS (Page 796)
MonDiS operating modes (Page 799)
MonDiS functions (Page 800)
MonDiS messaging (Page 804)
MonDiS I/Os (Page 805)
MonDiS block diagram (Page 808)

## 5.6.5 MonDiS messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Process messages
- Instance-specific messages

### Process control fault

You can interconnect an external fault (signal) to input parameter `CSF`. If it changes to `CSF` = 1, a process control fault is triggered (`MsgEvId1`, SIG 3).

### Process messages

You can use up to four instance-specific messages with this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId 1` | SIG 1 | Alarm - high | $$BlockComment$$<br>`Out` - Binary value set |
| | SIG 2 | Not used | |
| | SIG 3 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |
| | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External message 1 |
| | SIG 5 | AS process control message - fault | $$BlockComment$$<br>External message 2 |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance `MsgEvId1`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | `ExtVa104` |
| 5 | `ExtVa105` |
| 6 | Reserved |
| 7 | Reserved |
| 8 | Reserved |
| 9 | Reserved |
| 10 | Reserved |

The associated values 4 and 5 are allocated to the parameters `ExtVa104` and `ExtVa105` and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of MonDiS (Page 796)

MonDiS operating modes (Page 799)

MonDiS functions (Page 800)

MonDiS error handling (Page 803)

MonDiS block diagram (Page 808)

## 5.6.6 MonDiS I/Os

I/Os of MonDiS

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `AlmMsgEn` | 1 = Alarms are output | BOOL | 1 |
| `BatchEn` | 1 = Enable allocation | BOOL | 0 |
| `BatchID` | Batch ID | DWORD | 16#00000000 |
| `BatchName` | Batch name | S7-String | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| CSF | 1 = External error (control system fault)<br>Error handling (Page 803) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Color | Color for status display in the block icon:<br>0 = Default setting<br>1 = Red (alarm)<br>2 = Yellow (warning)<br>3 = Blue (tolerance)<br>4 = Black (control system message)<br>5 = Pastel green (process message)<br>6 = Violet (operator prompt) | BYTE | 16#00 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa104 | Associated value 4 for messages (MsgEvID1) | ANY | |
| ExtVa105 | Associated value 5 for messages (MsgEvID1) | ANY | |
| Feature | I/O for additional functions (Page 800) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| In | Digital input value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| InvIn | Invert input In | BOOL | 0 |
| MS_RelOp* | 1 = Release for maintenance by OS operator | BOOL | 0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 800) | STRUCT<br>• Bit 0: BOOL<br>• Bit 10: BOOL<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| Out_DC | Delay time for setting the output parameter Out for rising edges | REAL | 0.0 |
| RunUpCyc | Messages can be suppressed for a short time after start-up. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | 1 = Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SimOn | 1 = Simulation on | BOOL | 0 |
| SimIn* | Value used for SimOn = 1 | BOOL | 0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see MonDiS error handling (Page 803) | INT | -1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feed-forwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| Out | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 796) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 796) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 796) | DWORD | 16#00000000 |
| SumMsgAct | 1 = Group message is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### See also

MonDiS operating modes (Page 799)

MonDiS block diagram (Page 808)

Error handling (Page 128)

## 5.6.7 MonDiS block diagram

### MonDiS block diagram

A block diagram is not provided for this block.

### See also

Description of MonDiS (Page 796)
MonDiS operating modes (Page 799)
MonDiS functions (Page 800)
MonDiS messaging (Page 804)
MonDiS I/Os (Page 805)
MonDiS error handling (Page 803)

## 5.6.8 Operator control and monitoring

### 5.6.8.1 MonDiS views

#### Views of the MonDiS block

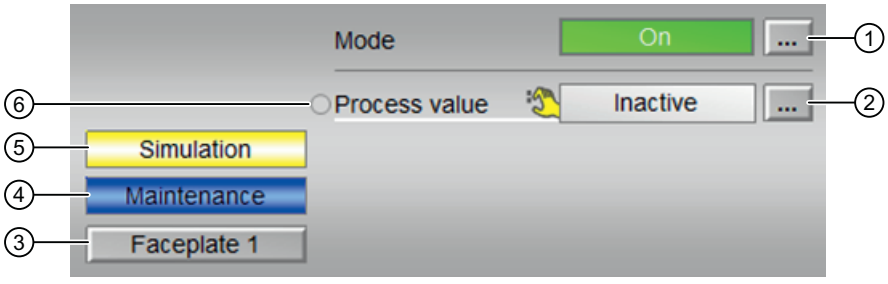
The block MonDiS provides the following views:

- MonDiS standard view (Page 809)
- Alarm view (Page 336)
- Trend view (Page 341)
- MonDiS parameter view (Page 811)
- MonDiS preview (Page 812)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for MonDiS (Page 813)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 5.6.8.2 MonDiS standard view

#### MonDiS standard view



### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 77)
- Out of service (Page 77)

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the operating mode.

### (2) Display of process value enabled/disabled

This area shows you the status of the individual connected parameters.

The identifier can be changed using Text 0 / Text 1 at the output parameter `Out`.

If the block is in simulation, you can enable or disable the process value. To do this, click on the display to open the operator input area.

If text is configure for these commands, it is displayed as additional text and as button labels for command selection. Additional information is available in the section Labeling of buttons and text (Page 235).

### (3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

Additional information is available in the section Opening additional faceplates (Page 234).

### (4) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

### (5) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"

### (6) Display of Inverted Input

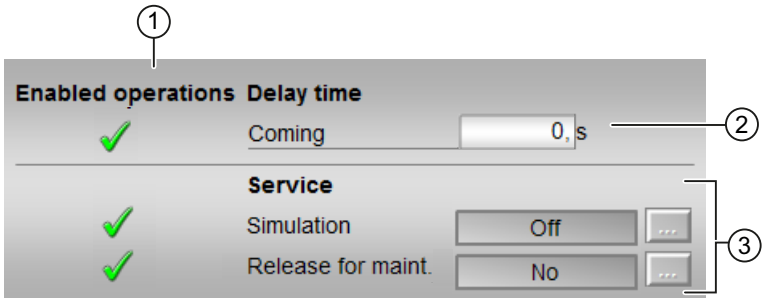
Blank circle will be displayed when input signal In is inverted with `InvIn`

### See also

Displaying auxiliary values (Page 239)

### 5.6.8.3 MonDiS parameter view

#### Parameter view of MonDiS




#### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

#### (2) Delay time

Enter here the delay time by which the output should be set. Enter delay times here for positive ("incoming", 0 - 1 edge) edges. Additional information is available in the section Switching operating states and operating modes (Page 289).

#### (3) Service

You can select the following functions in this area:

- "Simulation"
- "Release for maintenance"

Refer to the Switching operating states and operating modes (Page 289) section for more on this.

You can find information on this area in the section:

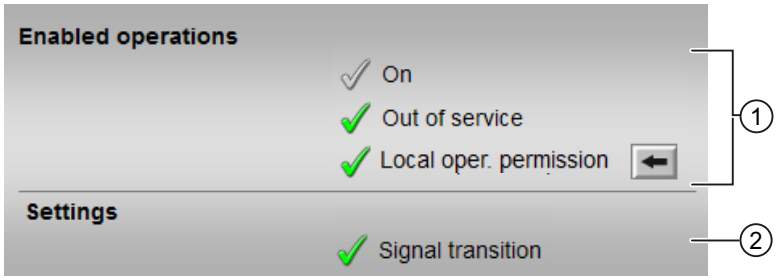
- Simulating signals (Page 64)
- Release for maintenance (Page 70)

### See also

Changing values (Page 291)

## 5.6.8.4 MonDiS preview

### Preview of MonDiS




### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. Additional information is available in the section Operator control permissions (Page 285).

### (2) Settings

- "Signal transition":
  - Activated: A message is generated with a "0 → 1" signal transition at the monitored input.
  - Deactivated: No message is generated.

### 5.6.8.5 Block icon for MonDiS

## Block icons for MonDiS

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as control system faults
- Operating modes
- Signal status, release for maintenance
- Memo display
- Display of the output signal
- Display configured instance-specific text for the process value (only for block icons 2 to 5). The instance-specific text for the process value can be changed using Text 0 / Text 1 at the output parameter `Out`.

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | The text background color for output `Out = 1` depends on the parameter assignment of the input "`Color`". |
|  | 3 | See 2 |

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 4 | The text background color for output `Out = 1` depends on the parameter assignment of the input "`Color`".<br>Text in block icon can be configured via Text 1 at the `Out` output parameter, a maximum of two characters are supported. |
|  | 5 | The text background color for output `Out = 1` depends on the parameter assignment of the input "Color". For `Color = 11`, the text background color is directly derived from EventState (but without the message classes process message and operator prompt) which also takes into consideration the CSF input.`Color = 11` is only supported by type 5. |

**Note**

Text 1 is also used in the standard view of the faceplate in the display process value active/inactive. Long static texts must therefore be displayed on the "Process value" text using the "`FlutTmIn`" parameter (see MonDiS standard view (Page 809))

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | The text background color for output `Out = 1` depends on the parameter assignment of the input "`Color`". |
|  | 3 | See 2 |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269).

# 5.7 MonDi08 - Monitoring 8 digital process tags

## 5.7.1 Description of MonDi08

### Object name (type + number) and family

Type + number: FB 1847

Family: Monitor

### Area of application for MonDi08

The block is used for the following applications:

- Monitoring of up to eight digital process tags

### How it works

The MonDi08 block is used to monitor up to eight digital process tags with flutter suppression.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

For the MonDi08 block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Monitoring eight digital process tags (Digital8Monitoring) (Page 2853)

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section MonDi08 I/Os (Page 824).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | Not used |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |

| Status bit | Parameter |
|---|---|
| 5 | Not used |
| 6 | OnAct.Value |
| 7 | Out1.Value |
| 8 | Out2.Value |
| 9 | Out3.Value |
| 10 | Out4.Value |
| 11 | Out5.Value |
| 12 | Out6.Value |
| 13 | Out7.Value |
| 14 | Out8.Value |
| 15 - 19 | Not used |
| 20 | In1 is used |
| 21 | In2 is used |
| 22 | In3 is used |
| 23 | In4 is used |
| 24 | In5 is used |
| 25 | In6 is used |
| 26 | In7 is used |
| 27 | In8 is used |
| 28 - 31 | Not used |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | MsgLock.Value |
| 1 | Alm1MsgEn |
| 2 | Alm2MsgEn |
| 3 | Alm3MsgEn |
| 4 | Alm4MsgEn |
| 5 | Alm5MsgEn |
| 6 | Alm6MsgEn |
| 7 | Alm7MsgEn |
| 8 | Alm8MsgEn |
| 9 | InvIn1 |
| 10 | InvIn2 |
| 11 | InvIn3 |
| 12 | InvIn4 |
| 13 | InvIn5 |
| 14 | InvIn6 |
| 15 | InvIn7 |
| 16 | InvIn8 |
| 17 - 28 | Not used |

| Status bit | Parameter |
|---|---|
| 29 | 1= Suppressed messages |
| 30 | Internal error |
| 31 | MS_RelOp |

### Status word allocation for Status3 parameter

| Status bit | Parameter |
|---|---|
| 0 | Effective signal 1 of the message block connected via EventTsIn |
| 1 | Effective signal 2 of the message block connected via EventTsIn |
| 2 | Effective signal 3 of the message block connected via EventTsIn |
| 3 | Effective signal 4 of the message block connected via EventTsIn |
| 4 | Effective signal 5 of the message block connected via EventTsIn |
| 5 | Effective signal 6 of the message block connected via EventTsIn |
| 6 | Effective signal 7 of the message block connected via EventTsIn |
| 7 | Effective signal 8 of the message block connected via EventTsIn |
| 8 - 15 | Effective signal 9..16 of the message block connected via EventTsIn |
| 16 - 31 | Not used |

### See also

MonDi08 messaging (Page 823)
MonDi08 functions (Page 819)
MonDi08 block diagram (Page 829)
MonDi08 error handling (Page 822)
MonDi08 modes (Page 818)

## 5.7.2 MonDi08 modes

### MonDi08 operating modes

This block provides the following modes.

- On (Page 77)
- Out of service (Page 77)

### "On"

You can find general information about the "On" mode in the On (Page 77) section.

### Out of service

You can find general information about the "Out of service" mode in the Out of service (Page 77) section.

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit5` =1 and `Feature.Bit5` =0 is as shown below:

| Output Parameters | `Feature.Bit5` =0 | `Feature.Bit5` =1 |
|---|---|---|
| Out1.ST | 16#80 | 16#60 |
| Out2.ST | 16#80 | 16#60 |
| Out3.ST | 16#80 | 16#60 |
| Out4.ST | 16#80 | 16#60 |
| Out5.ST | 16#80 | 16#60 |
| Out6.ST | 16#80 | 16#60 |
| Out7.ST | 16#80 | 16#60 |
| Out8.ST | 16#80 | 16#60 |

### See also

MonDi08 block diagram (Page 829)

MonDi08 I/Os (Page 824)

MonDi08 messaging (Page 823)

MonDi08 error handling (Page 822)

MonDi08 functions (Page 819)

Description of MonDi08 (Page 816)

## 5.7.3 MonDi08 functions

### Functions of MonDi08

The functions for this block are listed below.

### Monitoring and output of digital signals

The block is operated as "flutter filter". The block receives digital signals at the input `Inx` (x = 1 ... 8) which ideally do not develop any flutter. The block monitors these signals. Use the `FlutTmInx` (x = 1 ... 8) input to determine the duration of a continuous signal in order to transfer it to the process without flutter.

The preprocessed signal is sent to the process via the `Outx` (x = 1 ... 8) output.

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `In1.ST`
- `In2.ST`
- `In3.ST`
- `In4.ST`
- `In5.ST`
- `In6.ST`
- `In7.ST`
- `In8.ST`

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) section. The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 5 | Changes signal status of outputs in OOS (Page 208) |
| 11 | Suppress MsgLock and "Out of service" mode for a connected message block (Page 202) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 24 | Enabling local operator authorization (Page 169) |

### Operator control permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 - 11 | Not used |
| 12 | 1 = Operator can activate the Release for maintenance function |
| 13 | 1 = Operator can change the time for flutter suppression at the `In1` input |
| 14 | 1 = Operator can change the time for flutter suppression at the `In2` input |
| 15 | 1 = Operator can change the time for flutter suppression at the `In3` input |
| 16 | 1 = Operator can change the time for flutter suppression at the `In4` input |
| 17 | 1 = Operator can change the time for flutter suppression at the `In5` input |
| 18 | 1 = Operator can change the time for flutter suppression at the `In6` input |
| 19 | 1 = Operator can change the time for flutter suppression at the `In7` input |
| 20 | 1 = Operator can change the time for flutter suppression at the `In8` input |
| 21 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Changing labels on buttons and text

This block provides the standard function Labeling of buttons and text (Page 235).

You can change the label for "Process value" for this block as you please. The change is made with the `FlutXTmIn` parameter.

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

MonDi08 messaging (Page 823)

MonDi08 I/Os (Page 824)

Description of MonDi08 (Page 816)

MonDi08 block diagram (Page 829)

MonDi08 error handling (Page 822)

MonDi08 modes (Page 818)

EventTs functions (Page 2046)

Suppression of all messages (Page 185)

## 5.7.4 MonDi08 error handling

### Error handling of MonDi08

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Flutter alarm

### Overview of error numbers

The `ErrorNum` I/O can be used to output various error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 11 | Fluttering time < 0 |

### Flutter alarm

An alarm is output at output `FlutAct` with 1 if signal flutter is detected. Refer to the functions of the block > Monitoring and reporting flutter signals (Page 819).

### See also

MonDi08 block diagram (Page 829)

MonDi08 I/Os (Page 824)

MonDi08 messaging (Page 823)

MonDi08 modes (Page 818)
Description of MonDi08 (Page 816)

## 5.7.5 MonDi08 messaging

### Messaging

The following messages can be generated for this block:

- Process messages
- Instance-specific messages

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId 1 | SIG 1 | Alarm - high | $$BlockComment$$ Signal change Signal 1 has occurred |
| | SIG 2 | Alarm - high | $$BlockComment$$ Signal change Signal 2 has occurred |
| | SIG 3 | Alarm - high | $$BlockComment$$ Signal change Signal 3 has occurred |
| | SIG 4 | Alarm - high | $$BlockComment$$ Signal change Signal 4 has occurred |
| | SIG 5 | Alarm - high | $$BlockComment$$ Signal change Signal 5 has occurred |
| | SIG 6 | Alarm - high | $$BlockComment$$ Signal change Signal 6 has occurred |
| | SIG 7 | Alarm - high | $$BlockComment$$ Signal change Signal 7 has occurred |
| | SIG 8 | Alarm - high | $$BlockComment$$ Signal change Signal 8 has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance `MsgEvId1`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | `ExtVa104` |
| 5 | `ExtVa105` |
| 6 | `ExtVa106` |
| 7 | `ExtVa107` |
| 8 | `ExtVa108` |
| 9 | Reserved |
| 10 | Reserved |

The associated values 4 ... 8 are allocated to the parameters `ExtVa104` ... `ExtVa108` and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of MonDi08 (Page 816)
MonDi08 functions (Page 819)
MonDi08 I/Os (Page 824)
MonDi08 block diagram (Page 829)
MonDi08 error handling (Page 822)
MonDi08 modes (Page 818)
Time stamp (Page 231)

## 5.7.6 MonDi08 I/Os

I/Os of MonDi08

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `Alm1MsgEn` | 1 = Activate error message | BOOL | 1 |
| `Alm2MsgEn` | 1 = Activate error message | BOOL | 1 |
| `Alm3MsgEn` | 1 = Activate error message | BOOL | 1 |
| `Alm4MsgEn` | 1 = Activate error message | BOOL | 1 |
| `Alm5MsgEn` | 1 = Activate error message | BOOL | 1 |
| `Alm6MsgEn` | 1 = Activate error message | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Alm7MsgEn | 1 = Activate error message | BOOL | 1 |
| Alm8MsgEn | 1 = Activate error message | BOOL | 1 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkTypeData* | This input can not be edit or linked and is for internal usage only! | | |
| EN | 1 = Called block will be processed | BOOL | 1 |
| EventTsIn | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| ExtVa104 | Associated value 4 for messages (MsgEvID1) | ANY | |
| ExtVa105 | Associated value 5 for messages (MsgEvID1) | ANY | |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| ExtVa108 | Associated value 8 for messages (MsgEvID1) | ANY | |
| Feature | I/O for additional functions (Page 819) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| Flut1TmIn | Flutter time [s] | REAL | 0.0 |
| Flut2TmIn | Flutter time [s] | REAL | 0.0 |
| Flut3TmIn | Flutter time [s] | REAL | 0.0 |
| Flut4TmIn | Flutter time [s] | REAL | 0.0 |
| Flut5TmIn | Flutter time [s] | REAL | 0.0 |
| Flut6TmIn | Flutter time [s] | REAL | 0.0 |
| Flut7TmIn | Flutter time [s] | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Flut8TmIn | Flutter time [s] | REAL | 0.0 |
| In1 | Binary input In1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In2 | Binary input In2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In3 | Binary input In3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In4 | Binary input In4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In5 | Binary input In5 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In6 | Binary input In6 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In7 | Binary input In7 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In8 | Binary input In8 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| InvIn1 | Invert input In1 | BOOL | 0 |
| InvIn2 | Invert input In2 | BOOL | 0 |
| InvIn3 | Invert input In3 | BOOL | 0 |
| InvIn4 | Invert input In4 | BOOL | 0 |
| InvIn5 | Invert input In5 | BOOL | 0 |
| InvIn6 | Invert input In6 | BOOL | 0 |
| InvIn7 | Invert input In7 | BOOL | 0 |
| InvIn8 | Invert input In8 | BOOL | 0 |
| MS_RelOp* | 1 = Release for maintenance by OS operator | BOOL | 0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 819) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | 1 = Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SelFp2 | 1 = Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | - |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see MonDi08 error handling (Page 822) | INT | -1 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message acknowledgement status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | 1 = Alarm error 1 (output ERROR of the first ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| Out1 | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Out2 | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Out3 | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Out4 | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Out5 | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Out6 | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Out7 | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Out8 | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 816) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 816) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 816) | DWORD | 16#00000000 |

**See also**

MonDi08 messaging (Page 823)
MonDi08 block diagram (Page 829)
MonDi08 modes (Page 818)

## 5.7.7 MonDi08 block diagram

### MonDi08 block diagram

A block diagram is not provided for this block.

**See also**

MonDi08 I/Os (Page 824)
MonDi08 messaging (Page 823)
MonDi08 error handling (Page 822)
MonDi08 functions (Page 819)
MonDi08 modes (Page 818)
Description of MonDi08 (Page 816)

## 5.7.8 Operator control and monitoring

### 5.7.8.1 MonDi08 views

### Views of the MonDi08 block

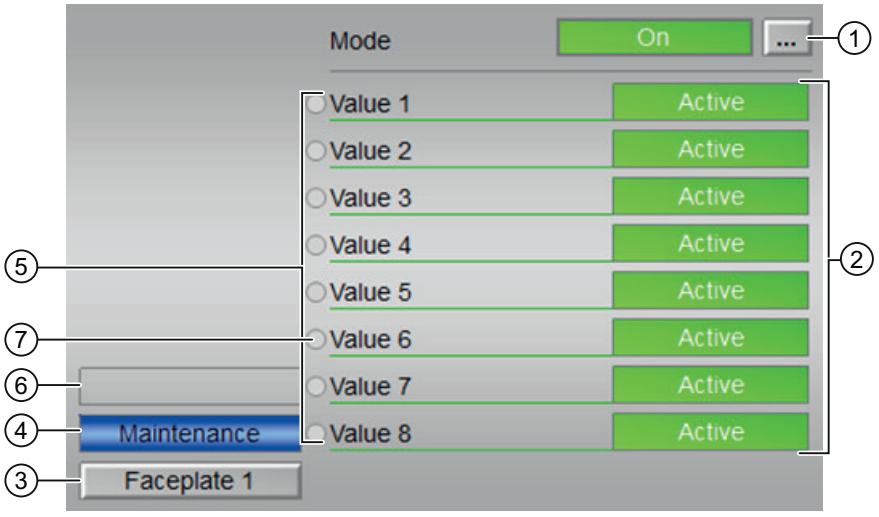
The block MonDi08 provides the following views:

- MonDi08 standard view (Page 830)
- Alarm view (Page 336)
- Trend view (Page 341)
- MonDi08 parameter view (Page 832)
- MonDi08 preview (Page 833)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for MonDi08 (Page 834)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 5.7.8.2 MonDi08 standard view

#### MonDi08 standard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 77)
- Out of service (Page 77)

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the operating mode.

#### (2) Display of the status for each parameter

This display is only visible when the corresponding block input is connected.

This area shows you the status of the individual parameters available.

You can determine the names for the connected parameters using the `S7_String` attribute at the corresponding input parameter. A default text is displayed if you enter nothing here.

You can change the text for value 1 ... 8 with the `FlutXTmIn` parameter.

Additional information is available in the section Labeling of buttons and text (Page 235).

#### (3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Opening additional faceplates (Page 234) section.

### (4) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

### (5) Displaying input values

**Changing the display:**

Follow these steps to change the displays:

1. In the CFC, right-click the MonDi08 block and select "Object Properties" from the context menu.
2. Select "I/Os" tab in the properties window.
3. In the “OS additional text” column, change the default setting for the input parameter (`FlutXTmIn`, where x = 1...8) to what you want to see during runtime later.
4. In the "Identifier" column, change the default setting for the input parameter (`FlutXTmIn`, where x = 1...8) to what you want to see during runtime later.
5. If both “OS additional text” and “Identifier” columns are empty, the default value is displayed in the runtime.
6. The text is used as a label and is therefore always displayed, which means it is independent of the signal status of the corresponding input (`FlutXTmIn`, where x = 1...8).
7. The font size is reduced during the runtime if the input text length is greater than the label width.

**Note**

If a text is added to the "OS additional text" field of the input parameter (`FlutXTmIn`, where x = 1...8), this text is displayed even if the "Identifier" text field of the same input parameter is not empty. If the "OS additional text" field of the input parameter (`FlutXTmIn`, where x = 1...8) is empty, the "Identifier" text of the same input parameter is displayed.

### (6) Display area for block states

This area provides additional information on the operating state of the block:

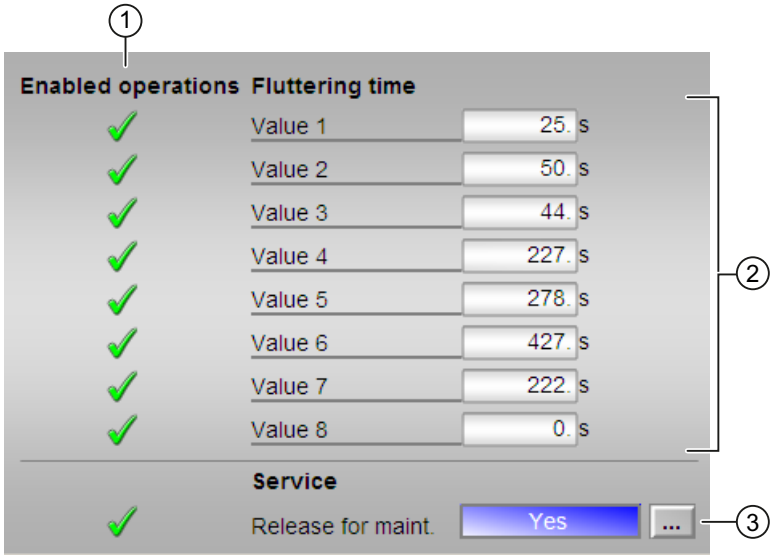
- Internal error

### (7) Display of Inverted Input

Blank circle will be displayed when input signal In is inverted with `InvIn`

### 5.7.8.3 MonDi08 parameter view

#### Parameter view of MonDi08




#### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

#### (2) Area for entering the flutter time

Use this area to set the time period to determine how long a continuous signal should last in order for it to be transferred to the process without flutter.

Refer to the Changing values (Page 291) section for more on this.

You can determine the names for the connected parameters using the `S7_String` attribute at the corresponding input parameter. A default text is displayed if you enter nothing here.

You can change the text for value 1 ... 8 with the `FlutXTmIn` parameter.

Additional information is available in the section Labeling of buttons and text (Page 235).

### (3) Service

You can select the following function in this area:

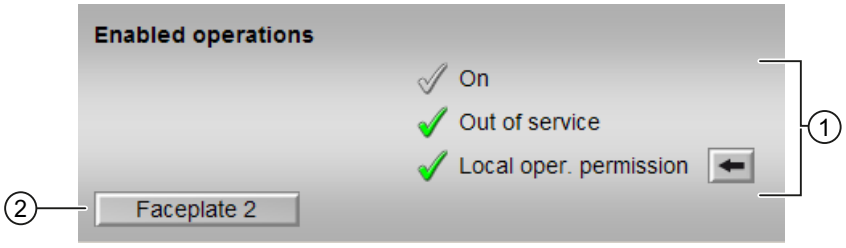
- "Release for maintenance"

Refer to the Switching operating states and operating modes (Page 289) section for more on this.

You can find information on this area in the Release for maintenance (Page 70) section.

## 5.7.8.4 MonDi08 preview

### Preview of MonDi08




### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. Additional information is available in the section Operator control permissions (Page 285).

### (2) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Opening additional faceplates (Page 234) section.

### 5.7.8.5 Block icon for MonDi08

#### Block symbols for MonDi08

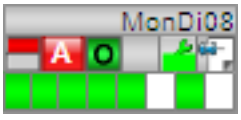
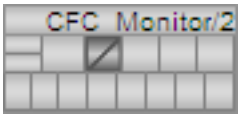
A variety of block symbols are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as control system faults
- Operating modes
- Signal status, release for maintenance
- Memo display
- Display of the output signal

The block symbols from template @TemplateAPLV9.PDL:

| Symbols | Selection of the block symbol in CFC | Special features |
|---|---|---|
|  | 1 | |

The block symbols from template @TemplateAPLV7.PDL:

| Symbols | Selection of the block symbol in CFC | Special features |
|---|---|---|
|  | 1 | Block symbol in the full display |
|  | 2 | |
|  | - | Block symbol in "Out of service" mode (example with type 1 block symbol) |

Additional information on the block symbol and the control options in the block symbol is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269).

# Controller blocks

# 6

## 6.1 Comparison of large & small blocks

### 6.1.1 PIDConL compared to PIDConS

#### Comparison of the PIDConL and PIDConS blocks

The following tables are intended to help you decide which block to use.

#### Memory and runtime savings of the small block compared to the large block

You save the following resources for each instance:

- Memory space: ~ 45%
- Runtime: ~ 40%

#### Block operating modes

| | PIDConL | PIDConS |
|---|---|---|
| Manual mode (Page 79) | X | X |
| Automatic mode (Page 79) | X | X |
| Program mode for controllers (Page 85) | X | |
| Out of service (Page 77) | X | X |

#### Functions of the blocks

| | PIDConL | PIDConS |
|---|---|---|
| Generating and limiting the manipulated variable | X | X |
| Tracking and limiting a manipulated variable (Page 222) | X | X |
| Tracking and limiting a manipulated variable (Page 222) | X | X |
| Neutral position for motors, valves and controllers (Page 54) | X | X |
| Outputting group errors (Page 130) | X | X |
| Outputting a signal for start readiness (Page 59) | X | X |

| | PIDConL | PIDConS |
|---|---|---|
| "Actuator active" information | X | X |
| Limit monitoring of the feedback (Page 101) | X | |
| Setpoint specification - internal/ external (Page 136) | X | X |
| Setpoint limiting for external setpoints (Page 222) | X | X |
| Gradient limit of the setpoint (Page 132) | X | |
| Using setpoint ramp (Page 131) | X | |
| Tracking setpoint in manual mode (Page 222) | X | X |
| Simulating signals (Page 64) | X | X |
| Limit monitoring of the process value (Page 93) | X | X |
| Alarm delays with two time values per limit pair | | |
| Alarm delay with one time value per limit pair | X | X |
| Error signal generation and dead band (Page 217) | X | X |
| Delay alarm for control deviation at setpoint step changes (Page 216) | X | |
| Limit monitoring of error signal | X | |
| Inverting control direction (Page 217) | X | |
| Physical standardization of setpoint, manipulated variable and process value | X | |
| Selecting a unit of measure (Page 240) | X | X |
| PID algorithm | X | X |
| Structure segmentation at controllers (Page 224) | X | |
| Anti-windup (Page 1024) | X | X |
| Feedforwarding and limiting disturbance variables (Page 223) | X | |
| Using control zones (Page 220) | X | |
| Forming and outputting the signal status for technologic blocks (Page 117) | X | X |
| Operator control permissions (Page 285) | X | X |
| Release for maintenance (Page 70) | X | X |
| Generating instance-specific messages (Page 230) | X | X |

| | PIDConL | PIDConS |
|---|---|---|
| Suppressing messages using the MsgLock parameter (Page 231) | X | X |
| Display and operator input area for process values and setpoints (Page 233) | X | X |
| Opening additional faceplates (Page 234) | X | X |
| SIMATIC BATCH functionality (Page 73) | X | X |
| Time stamp | X | |
| Dead band is temporarily disabled (Page 152) | X | |
| Labeling of buttons and text (Page 235) | X | |
| Displaying current control signals from output channel block | X | X |
| Interlocks (Page 107) | X | |
| Disabling interlocks (Page 111) | X | |

## Configurable functions using the `Feature` parameter

| Bit number | `Feature` bit function | PIDConL | PIDConS |
|---|---|---|---|
| 0 | Set startup characteristics (Page 148) | X | X |
| 1 | Reaction to the out of service mode (Page 188) | X | X |
| 2 | Resetting the commands for changing the mode (Page 172) | X | X |
| 4 | Setting switch or button mode (Page 177) | X | |
| 12 | Control zone with specified I component (Page 171) | X | |
| 13 | Control zone with frozen I component (Page 171) | X | |
| 14 | External control deviation (Page 161) | X | |
| 15 | Neutral position manipulated variable takes effect with "out of service" operating mode (Page 177) | X | X |
| 16 | Neutral position manipulated variable takes effect at startup (Page 176) | X | X |
| 18 | Disabling bumpless switchover to automatic mode for controllers (Page 183) | X | |
| 22 | Update acknowledgment and error status of the message call (Page 171) | X | |
| 24 | Enabling local operator authorization (Page 169) | X | X |

| Bit number | `Feature` bit function | PIDConL | PIDConS |
|---|---|---|---|
| 25 | Suppression of all messages (Page 185) | X | X |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) | X | X |
| 28 | Disabling operating points (Page 156) | X | |
| 29 | Signaling limit violation (Page 181) | X | |
| 30 | Dead band is temporarily disabled (Page 152) | X | |

# 6.2 ConPerMon - Monitoring of the control performance of control loops

## 6.2.1 Description of ConPerMon

### Object name (type + number) and family

Type + number: FB 1805

Family: Control

### Area of application for ConPerMon

The block is used for the following applications:

- Permanent monitoring of control performance of control loops for early detection of problems as they develop

The block calculates:

- Stochastic characteristics of the control performance with the process in a steady state
  - Mean value, variance and standard deviation of controlled variable
  - Mean value of the manipulated variable and control deviation
  - Control performance index
  - Estimated steady state process gain
- Deterministic characteristics of the control performance with step changes in the setpoint
  - Response time and settling time and the settling ratio
  - Overshoot absolute and relative to the step height

Other statistical and graphic evaluations of the signals in the control loop over longer, freely selectable periods are available in the faceplate of the ConPerMon block.

In an overview representation of a plant or unit, you can obtain a clear picture of the status of all control loops based on ConPerMon block icons (indicator light function).

The aim is to detect problems as they develop and to focus the attention of the user on the control loops in a plant that are no longer operating correctly.

### How it works

The ConPerMon block evaluates the setpoint and process value signals and the manipulated variable of the PID controller in a sliding time window. The mode of the controller is also taken into account.

With the process in a steady state, the detected stochastic characteristics are compared with the reference values obtained during commissioning. If there is a step change in the setpoint, the stochastic characteristics are by definition irrelevant and are temporarily frozen. Instead, the monitoring of the deterministic characteristics is automatically activated.

If the control performance falls below a defined limit a message is generated. This is also the case when a defined limit for overshoot is exceeded when there is a step change in the setpoint.

## Configuration

Each PID controller has a ConPerMon block assigned to it that is installed in the same CFC chart and interconnected with the controller. This already takes place with the corresponding process tag types.

You can open the standard view for the ConPerMon block from the standard view of a controller (for example PIDConL). Additional information on this topic is available in the section Opening additional faceplates (Page 234).

After successful commissioning and optimization of the PID controller to be monitored, the ConPerMon block is initialized while the process is in a steady state and it stores the corresponding characteristic values as reference values.

Follow the steps outlined below:

- Change the PID controller you want to monitor to automatic mode and set the setpoint to the typical operating point. This operating status is intended to represent the normal operation of the process; in other words, the entire plant/unit should be running under production conditions. Monitor the process with a trend writer (CFC trend in the Engineering-System or WinCC Online-Trend-Control on the Operator Station) and wait until the process has settled
- To specify the length `TimeWindow` of the sliding time window, monitor the `PV_Variance` block output of the ConPerMonblock in a trend. The time window should be long enough to keep the variance fairly constant in the relevant decimal places. If the selected time window is too short in relation to the time constants in the control loop and the disturbance signal spectrum, the variance will have too much noise and no useful information.
  If the selected time window is too long, it takes longer before any deterioration of the control performance is detected by the ConPerMon block. It also takes longer following a step change in the setpoint before the monitoring of the stochastic characteristics can be resumed. A good starting value for the `TimeWindow` parameter is 10 times as long as the longest process time constant or 20 times as long as the reset time of the PID controller.
- If the controller
  - is set perfectly,
  - has achieved a steady state,
  - the time window has been defined and filled with values from the steady state,

  the ConPerMon block can be initialized. You do this by clicking the "Initialize" button in the parameter view of the ConPerMon faceplate or by setting the `InitRefVar` = 1 parameter in the CFC block. This saves the `PV_Variance` parameter in the current time window as a reference value for calculating the control performance in the block along with reference values for manipulated variable and process variable.

The **C**ontrol **P**erformance **I**ndex CPI should now be approximately at 100% and therefore indicate that the control loop is operating correctly. Due to stochastic fluctuations, the CPI can also temporarily exceed the 100% mark. If, however, the CPI drops by a significant amount over a longer period, this indicates deterioration of the control performance.

For more detailed information on interpreting the calculation results of the block, refer to the section ConPerMon functions (Page 845).

**Note**

If the length of the time window is changed during runtime, the CPI will temporarily deviate considerably from its old value and then gradually settle to the new steady value. It is advisable to reinitialize the ConPerMon block after the CPI value has settled to a constant level.

The ConPerMon faceplate is opened from the faceplate of the assigned PID controller so that the ConPerMonicon pictures do not need to be installed separately in each OS picture. It is, in fact, advisable to group all ConPerMon block icons of a plant or unit in one overview picture at the appropriate hierarchy level.

You can expand this overview picture with the trend display of the control performance of all control loops over a longer time to allow you to recognize gradual deterioration (for example reflecting wear and tear). You can also display a further view of the message archive (WinCC AlarmLogging Control) as a hit list sorted according to the frequency in which they occur. In this list, the control loops that caused the most alarms will be shown at the top.

For the ConPerMon block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 2827)
- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 2825)
- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 2822)
- PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 2823)
- Ratio control with control loop monitoring through ConPerMon (RatioControl) (Page 2836)
- Ratio control with PIDConR (RatioR) (Page 2839)
- PID controller with Smith predictor (SmithPredictorControl) (Page 2830)
- Step controller with direct access to the actuator and without position feedback (StepControlDirect) (Page 2831)

## Startup characteristics

When the CPU starts up, the block is reinitialized but the stored reference values are retained. The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

**Note**

For ConPerMon, the default value of `RunUpCyc` is set very high (32000). You must adapt it to your own process by taking into account the freely selectable signal inputs `ExtMsg1..3`.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section ConPerMon I/Os (Page 860)

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | Not used |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `PID_AutAct.Value` |
| 6 | `OnAct.Value` |
| 7 | Not used |
| 8 | Internal error |
| 9-14 | Not used |
| 15 | `CPI_Suppress` |
| 16 - 31 | Not used |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `MsgLock` |
| 1 | `OvsAH_Act.Value` |
| 2 | `OvsWH_Act.Value` |
| 3 - 4 | Not used |
| 5 | `CPI_WL_Act.Value` |
| 6 | `CPI_AL_Act.Value` |
| 7 | `OvsAH_En` OR `Feature.Bit31` |
| 8 | `OvsWH_En` OR `Feature.Bit31` |
| 9 - 10 | Not used |
| 11 | `CPI_WL_En` OR `Feature.Bit31` |
| 12 | `CPI_AL_En` OR `Feature.Bit31` |
| 13 | `OvsAH_MsgEn` |
| 14 | `OvsWH_MsgEn` |
| 15 - 16 | Not used |
| 17 | `CPI_WL_MsgEn` |
| 18 | `CPI_AL_MsgEn` |
| 19 | Delay of the `CPI_WL_Lim` message |
| 20 | Delay of the `CPI_AL_Lim` message |
| 21 | Collection of message delays |
| 22 - 23 | Not used |
| 24 | 1= Suppressed messages |
| 25 - 31 | Not used |

**See also**

ConPerMon messaging (Page 857)

ConPerMon block diagram (Page 866)

ConPerMon error handling (Page 857)

ConPerMon modes (Page 845)

## 6.2.2 ConPerMon modes

### ConPerMon operating modes

The block can be operated using the following modes

- On (Page 77)
- Out of service (Page 77)

### "On"

You can find general information about the "On" mode in the On (Page 77) section.

### "Out of service"

You can find general information about the "Out of service" mode in the Out of service (Page 77) section.

**See also**

ConPerMon block diagram (Page 866)

ConPerMon I/Os (Page 860)

ConPerMon messaging (Page 857)

ConPerMon error handling (Page 857)

ConPerMon functions (Page 845)

Description of ConPerMon (Page 841)

## 6.2.3 ConPerMon functions

### Functions of ConPerMon

The functions for this block are listed below.

### Monitoring of stochastic characteristics of the control performance

The mean value of a variable relating to an ergodic stochastic process (Page 2886) can be determined from a sliding time window with the length n = `TimeWindow` / `SampleTime`, for example, for the controlled variable y = `PV`:

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y(i)$$

A recursive formulation of this calculation is included in the MeanTime block that is called by the ConPerMonblock. Most steady-state time series can be considered as being ergodic so that the expected value can be estimated by averaging over a window of finite length.

The mean control deviation is `ER_Mean` = `SP` - `PV_Mean`. A mean steady-state control deviation ≠ 0 at a constant setpoint is an indication of problems in the control loop if the controller has I component. You should then check the following potential causes:

- The actuator does not have sufficient capacity. As a result, the controller's actuating signal constantly approaches its limit. This can be caused by unsuitably dimensioned actuators or may simply by wear and tear.
- The manipulated variable demanded by the controller does not take effect in the process, for example because the actuator is defective.

If a steady-state reference operating point (`MV_Ref`, PV_Ref) is known, this can be used to estimate the current mean steady state gain of a linear process model if it is assumed that only disturbances without mean value have an effect:

$$\mathrm{StatGain} = \frac{\mathrm{PV}_{\mathrm{Mean}} - \mathrm{PV}_{\mathrm{Ref}}}{\mathrm{MV}_{\mathrm{Mean}} - \mathrm{MV}_{\mathrm{Ref}}}$$

Normally the reference operating point is obtained during the initialization of the ConPerMon block. Estimation of the steady state gain is then, however, impossible precisely at this operating point. As an alternative, you can also enter the reference values `PV_Ref` and `MV_Ref` manually at the appropriate block inputs. Typical steady-state operating points are often known in advance, for example

- Flow control: `PV` = 0 for `MV` = 0, in other words, valve closed,
- Temperature control: `PV` = `PV_Ambient` for `MV` = 0, in other words, ambient temperature

If the steady state gain changes gradually as time progresses, this is an indication of wear phenomena in the process, such as deposits on heat exchangers, valves or shutters, failing efficiency of process plant, etc.

If, for example, a temperature regulation circuit is closed by a heat exchanger and a deposit forms on the exchanger surfaces, the heat transfer coefficient, and consequently the process gain, is reduced. Within certain limits, this can be compensated by a closed control loop (so that the controller initially disregards the problem). Although the original control loop dynamics can be restored (to a certain extent) by suitable increase of the controller gain as the pollution increases, it is advisable to eliminate the cause of the problem; in other words, to clean the heat exchanger.

If the estimated steady state gain changes suddenly and temporarily, this tends to point to an external disturbance. This may be a normal occurrence in the operation of the process. If, however, these occurrences become more frequent, it is worth finding out the cause.

Due to the approach, the variance `PV_Variance` as second moment requires the calculation of differences of each current measured value from (constant !) mean value:

- `Feature.Bit8` = 0 (default), approximate calculation with reduced computing power consumption:

$$\sigma_y^2 = \frac{1}{n}\left(\sum_{i=1}^{n} y^2(i)\right) - \overline{y}^2$$

- `Feature.Bit8` = 1, more exact calculation corresponding to mathematical definition of variance:

$$\sigma_y^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(y(i) - \overline{y}\right)^2$$

The standard deviation

$$PV_StdDev = \sigma_y = \sqrt{\sigma_y^2}$$

as the square root of the variance is easier to interpret because it has the same physical unit as the measured value.

The control performance index CPI (**C**ontrol **P**erformance **I**ndex) in the unit [%] describes the current variance of the controlled variable relative to a reference variance (benchmark). It is defined as

$$\varsigma = \frac{\sigma_{ref}^2}{\sigma_y^2} 100\%$$

The CPI moves in the $0 < \varsigma \leq 100\%$ range. If the current variance corresponds to the benchmark, the index reaches the value 100. If, by contrast, the current variance becomes larger, the control performance index decreases accordingly. Ideally, the benchmark is obtained in a defined good state of the control loop and stored when the ConPerMon block is initialized. It does not matter if the CPI temporarily reaches values somewhat higher than 100%. A CPI > 100% only means that the variance of the controlled variable is currently somewhat lower than in the reference state. Other alternatives for determining the benchmark will be explained in a separate .

If you consider the calculated CPI signal to be too strongly affected by noise, you can smooth it using the integrated low pass filter (parameter `CPI_FiltFactor`) with the filter time constant `TimeWindow · CPI_FiltFactor`.

The disadvantage of these stochastic characteristics is that they assume an ergodic (Page 2886) or steady state in the process - at least in a statistical sense. Each step change in the standpoint in a controller is an elementary violation of this requirement and leads temporarily to incorrect statements of the stochastic characteristics, for example variances increasing too much. The basic principle of the combined approach implemented in the ConPerMonblock is to use both stochastic and deterministic characteristics for the control performance and to select the suitable characteristics automatically depending on the operating state.

If a step change in the setpoint is detected in a control loop, the ConPerMon block freezes the CPIvalue and automatically suppresses all messages relating to this. As a user, you can also force the suppression of the messages manually via the `ManSupprCPI` = 1 binary input. This setting is useful to avoid false alarms when known disturbances occur, for example at a load change in a

Conti process (Page 2887) or a dosing procedure in a Batch process (Page 2881). In such cases the variance of the controlled variable usually rises momentarily. This should not be interpreted as a worsening of the control performance.

### Monitoring of deterministic characteristics of the control performance

Assessment of the control performance based on the response to a step change in the setpoint is relatively simple. In the sense of automatic monitoring, the ConPerMon block is capable of determining the essential characteristics of the control performance directly from the signal changes so that when necessary a message or an alarm can be generated automatically by the system.

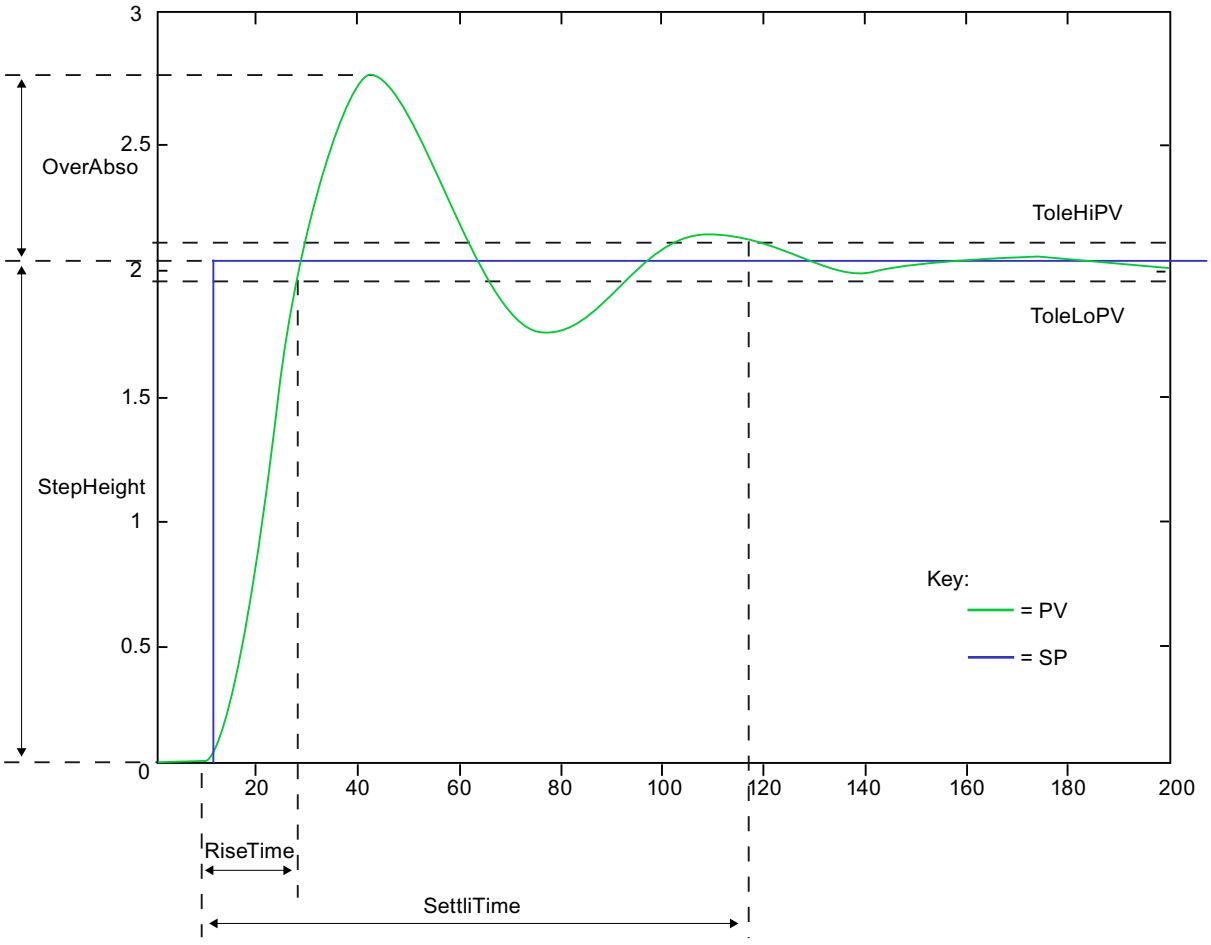
The first thing to look for is always the overshoot if it is present and clearly distinct from the noise level. For a positive step response,

`OverAbso` = max(`PV`) - `SP` > 0

is output where is for a negative step response (step response down), and negative values

`OverAbso` = min(`PV`) - `SP` < 0

are also output. For normalization, the absolute overshoot is related to the height of the step change in the setpoint and is therefore always positive. The relative overshoot (`Overshoot`) as a percentage is a measure of the damping of the control loop. If this is more than 20 or 30%, the loop gain (gain of the controller multiplied by the gain of the control system) is generally too high either because the controller was badly set from the beginning or because the properties of the control system have changed over the course of time. If overshoot is significantly too high, the control loop is generating weakly damped oscillations in the plant. The block sends a message to this effect if the relative overshoot is above a specified limit.




In every control loop, there is a general correlation between overshoot and phase reserve: The higher the overshoot, the lower the phase reserve. If the response of the closed control loop can be described approximately by a 2nd order transfer function

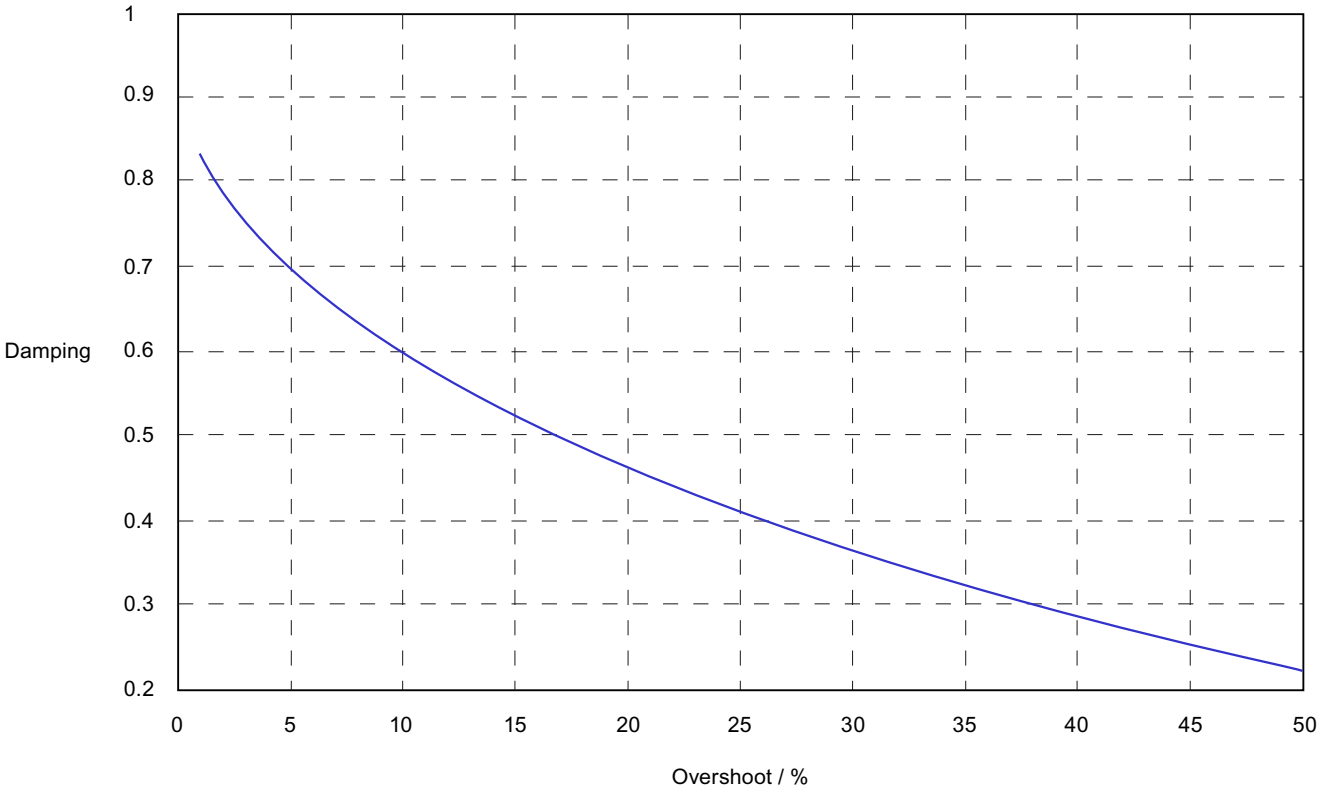
$$g_{cl}(s) = \frac{PV(s)}{SP(s)} = \frac{1}{\frac{1}{\omega_0^2}s^2 + 2\frac{\delta}{\omega_0}s + 1}$$

the following relationships are known:

- If $\delta \geq 1$ , the overshoot is equal to zero and the settling response is asymptotic.
- If $\delta < 1$ , overshoot and oscillations occur.

The damping of the closed loop can be determined approximately from the overshoot:

$$\delta = \frac{-\ln\left(\frac{\text{Overshoot}}{100\%}\right)}{\sqrt{\ln^2\left(\frac{\text{Overshoot}}{100\%}\right) + \pi^2}}$$




An optimum controller setting typically aims for overshoot between 5 and 25%, which means damping between 0.7 and 0.4.

If overshoot is too high, it is often helpful to reduce the gain of the controller.

While overshoot primarily serves to check controller gain, there is a further characteristic that provides information on the setting of the I component: If the setting of the reset time is unsuitable, the process value will creep towards the new setpoint following a step change in the setpoint. The ratio to the `RiseTime` is relevant, and not the absolute value of the `SettliTime`. If the settling ratio, in other words the quotient of the rise time and settling time, is less than approximately 25%, it can generally be assumed that the reset time of the controller is too slow. To determine the rise time and settling time, a 3σ tolerance band is placed around the setpoint and is also displayed in the faceplate of the ConPerMon block. The absolute values of the settling time and rise time can be assessed in terms of the concrete requirements of the process control for a specific application.

During a step change in the setpoint, larger mathematical variances of the controlled variable are bound to occur compared with the steady state so that generating alarms due to the variance limits being exceeded needs to be suppressed until the settling process has neared completion following the step change in the setpoint. The calculated deterministic characteristics are then output and the stochastic evaluation is reactivated.

### Group display SumMsgAct for limit monitoring, CSF and ExtMsgx

The block provides the standard function Group display for limit monitoring, CSF and ExtMsgx (Page 92).

### Alternatives for determining the benchmark

During planned commissioning of a plant with integrated ConPerMon, following controller optimization, the ConPerMon block is initialized for every control loop and the calculated variance stored as the benchmark for calculating the CPI.

As an alternative, a benchmark can be set via the `RefVarExt` input parameter by setting the `RefVarExtOn` = 1 input parameter. There are various ways of obtaining numeric values for the benchmark:

- Take the lowest variance that was ever measured in this control loop since the initialization of the ConPerMon block. This is displayed at the `PV_VarMin` output parameter. This value is only useful when the control loop has been in a stable and desirable operating state for a longer period of time at least once since the initialization of the ConPerMon block.
- Take the variance of the control loop with a theoretical minimum variance controller as can be obtained based on archived data using another supplier's CPM application. This depends only on the process dead time and the disturbance model. This form of CPI is known as the Harris index and represents a lower barrier that can generally not be reached by a PID controller which is why CPI seldom reaches the value 100% even by well tuned controllers. Low CPI values provide the first indication that the controller settings could be improved. You should, however, bear in mind that the minimum variance is only a theoretically achievable value and that the minimum variance controller has characteristics that are not desired in the real application, for example extremely high manipulated variable amplitudes. With minimum variance-based CPI, therefore, it is not worth making every effort to bring this as close as possible to 100%

### Cascade control

In a cascade control, you should only use the ConPerMon block for the primary controller and not for the secondary controller. The ConPerMon block cannot make any useful statements about the control performance of the secondary controller because

- the variance of the process value in the secondary control loop depends directly on the variance of the setpoint that is set as the manipulated variable by the primary controller,
- there are neither operating phases with a constant setpoint nor defined step changes in the setpoint.

Apart from this, from the perspective of process control, the primary control loop is, of course, the one whose control performance should be monitored while the control performance of the secondary loop is of secondary importance. It is nevertheless advisable to set the secondary controller carefully before optimization and monitoring of the primary controller is started because a poor response by the secondary controller cannot be compensated by the primary controller.

For additional information, read about the process tag template Cascade control with control loop monitoring through ConPerMon (CascadeControl) (Page 2840).

### Split-range control

The split-range function block contains two separate (static) characteristics for both actuators. Any significant difference between the two actuators in terms of performance (in other words, different steady state gains for heating and cooling) can be compensated by setting different gradients for the characteristics, so that the controller is presented with a linear process response (regardless of the sign) as far as possible. It this does not work, the control performance will differ slightly in the two areas. The initialization of the ConPerMon block should then be performed in the worse area to avoid error alarms.

For additional information, read about the process tag template Split-range controller with control loop monitoring through ConPerMon (SplitrangeControl) (Page 2833).

### PID controller with gain scheduler

The aim of gain scheduling is to achieve consistent control performance over the entire operating range. If this does not work perfectly, the initialization of the ConPerMon block should be performed in an operating point with worse control performance to avoid error alarms. We recommend that you expand the alarm limits somewhat at the ConPerMon block: permit lower `CPI`s and higher overshoot.

For additional information, read about the process tag template PID - control with operating-point-oriented parameter control (GainScheduling) (Page 2825).

### Override control

For change-over control, different controllers are active depending on the process state; their control performances differ, of course. We recommend using control loop monitoring only for the primary controller, and to suppress it using the `ManSuprCPI` input parameter if limit controlling is activated.

For additional information, read about the process tag template Override control (Page 2846).

### Feedforward control

The task of feedforward control is to avoid or at least to reduce degradation of the control performance caused by a measurable disturbance variable. Control loop monitoring therefore can basically be used as it is used for simple control loop. When the disturbance variable is quiet for a time and then acts up for a brief period, the resulting fluctuations of the control performance cannot be ruled out. The reason behind it is that feedforward control represents a model-based intervention, and a model is never a perfect reflection of reality.

For additional information, read about the process tag template PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 2827).

### Smith predictor

The Smith predictor enables higher control performance than a simple PI controller in control loops with dead time. Control loop monitoring therefore can basically be used as it is used for simple control loop. If the dead time changes during ongoing operation, control performance will most likely go down.

For additional information, read about the process tag template PID controller with Smith predictor (SmithPredictorControl) (Page 2830).

### Ratio control

With ratio controlling, the control loop monitoring should only be used in the primary control loop if the setpoints are to be determined for combined components from the actual value of the primary component. In this case, you can expect continuous setpoint changes in the control loops for the combined components - similar to sequential control loop of a cascade. If the setpoints for combined components are to be determined from the setpoint of the primary component, the lower-level control loops can be monitored as well.

For additional information, read about the process tag template Ratio control with control loop monitoring through ConPerMon (RatioControl) (Page 2836).

### Multivariable controller

The mathematical concept of the ConPerMon block is intended for monovariable control loops. If the variance in a control loop is found to be too high, the block cannot determine whether the actual cause is within the control loop or whether influences are being brought in due to interactions from the outside. If, therefore, you notice strong interactions between various control loops in your plant or even use multivariable controllers, the information provided by the ConPerMon block should be treated with caution.

It can nevertheless make sense to equip a multivariable controller such as the ModPreCon block with control loop monitoring to establish whether the control performance achieved during commissioning of the controller is also retained in runtime. In this case, each controller channel of the multivariable controller has a separate ConPerMon block. Several additional logic functions need to be configured upstream from the `ManSuprCPI` input parameter as shown in the corresponding sample project Predictive control of a 2x2 multi-variable controlled system (ModPreConSim) (Page 2879):

- If one or more other channels for the multivariable controller is in a non-steady state (for example, step change in the setpoint) indicated by the `CPI_SupRoot` = 1 output parameter, the temporarily increased variance cannot be avoided in this controller channel and should not cause a CPI message.
- If one or more other channels of the multivariable controller have higher variances (poor control performance) indicated by the corresponding output `CPI_WrnAct` = 1, due to the interaction, these variances also cause a higher variance in this controller channel that cannot be avoided and should not lead to a CPI warning. It is possible to find the actual cause of a disturbance in a multivariable system as follows: The channel that first detects higher variances, set the alarm while subsequent alarms in adjacent channels are suppressed.

  **Note**

  In the case of multivariables, the estimated steady state gains from the monovariable observation are irrelevant. By setting the input parameter `StGainValid` = 0, this status is also displayed in the faceplate as "Uncertain, process related".

If a PID controllers is remotely controlled in program mode (Page 85), it should be treated similar to a secondary controller for a cascade connection in regard to control performance monitoring, i.e. monitoring is usually impractical in this case.

If program mode is the typical operating mode of the controller involved, the corresponding ConPerMon block can be completely removed. If the controller involved is often used in automatic mode, however, monitoring can be temporarily disabled during program mode by connecting the output parameter `AdvCoAct` of the PIDConL block to the input parameter `ManSupprCPI` of the ConPerMon block.

### Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 240).

### Forming the signal status for blocks

The block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status for the block is formed from the following parameters:

- `SP_Mon.ST`
- `PV_Mon.ST`
- `MV_Mon.ST`
- `ER_Mean.ST`
- In addition, the ConPerMon block has the following special functions for determining status values:
- If you use a step controller without position feedback, there is no manipulated variable that you could interconnect with the input parameter `MV_Mon`. Unlike most other input parameters `MV_Mon`does not have the preset signal status "Uncertain, process related" (16#78). If there is no interconnected value, this status is transferred to the calculated output parameters `MV_Mean` and `StatGain`.
- `StGainValid`: Set this input to 0 if you use a multivariable controller or observe strong interactions between neighboring control loops. This gives the calculated output parameter `StatGain` the signal status "Uncertain, process related". If known disturbances affect your process, for example dosing procedures in a batch process, you can also set this input temporarily using the recipe control.
- Under normal circumstances the output parameter `StatGain` accepts the worse signal status of `PV_Mon` and `MV_Mon`. Other possible causes of uncertain status for `StatGain` are:
  - the process is currently very close to the reference operating point, or
  - the process is currently in transition, e.g. step change in the setpoint.
- The signal status of the CPI output parameter is dependent on output parameter `CPI_Suppress`: If `CPI_Suppress` = 1, the control performance index CPI is uncertain. Apart from this, the CPI can also become uncertain in occasional situations when there are numeric problems in the calculation of the variance. Under normal circumstances the CPI signal status is the same as the `PV_Mon` signal status.
- The signal status of the `OverAbso` output parameter is set to invalid when step changes in the setpoint are evaluated whose step change height is too low in relation to the noise level.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 1 | Reaction to the out of service mode (Page 188) |
| 8 | Variants of CPI calculation (Page 202) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 28 | Disabling operating points (Page 156) |
| 29 | Signaling limit violation (Page 181) |
| 31 | Limits disabled in faceplate (Page 206) |

### Configurable reactions using the `Feature2` parameter

| Bit | Function |
|---|---|
| 12 | Disable limit act output when limit value message is disabled (Page 209) |

### Operator permissions

The block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 - 1 | Not used |
| 2 | 1 = Operator can switch to "Out of service" mode |
| 3 - 27 | Not used |
| 28 | 1 = Operator can initialize the block |
| 29 | 1 = Operator may input a value for the time window, the reference value for the controlled variable and the reference value of the manipulated variable |
| 30 | 1 = Operator can abort the evaluation of the step response |
| 31 | Not used |

The block has the following permissions for the `OS1Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can change the limit (overshoot) for high alarm |
| 1 | 1 = Operator can change the limit (process value) for the high warning |
| 2 | Not used |
| 3 | 1 = Operator may change a value for the `CPI` hysteresis. |
| 4 | Not used |
| 5 | 1 = Operator can change the limit (control performance index `CPI`) for the low warning |

| Bit | Function |
|---|---|
| 6 | 1 = Operator can change the limit (control performance index CPI) for the low alarm |
| 7 | 1 = Operator can activate / deactivate messages via CPI_WL_MsgEn |
| 8 | 1 = Operator can activate / deactivate messages via CPI_AL_MsgEn |
| 9 | 1 = Operator can activate / deactivate messages via OvsAH_MsgEn |
| 10 | 1 = Operator can activate / deactivate messages via OvsWH_MsgEn |
| 7 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in OS_Perm as a parameter, you have to reset the corresponding OS_Perm bit.

### Alarm delays with a time value for all limits

The block provides the standard function One time value for all limits (Page 226).

This function is used only for monitoring the control performance index CPI.

### Limit operation and display in the faceplate

This block provides the standard function Limit operation and display in the faceplate (Page 355).

### Generating instance-specific messages

The block provides the standard function Generating instance-specific messages (Page 230) without the time stamp function in the I/O.

### Suppressing messages using the MsgLock parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of ConPerMon (Page 841)

ConPerMon messaging (Page 857)

ConPerMon I/Os (Page 860)
ConPerMon block diagram (Page 866)
ConPerMon error handling (Page 857)
ConPerMon modes (Page 845)

## 6.2.4 ConPerMon error handling

### Error handling of ConPerMon

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 2 | `SampleTime` < 0.001 |
| 10 | `TimeWindow` < 20 · `SampleTime` |

### See also

ConPerMon block diagram (Page 866)
ConPerMon I/Os (Page 860)
ConPerMon messaging (Page 857)
ConPerMon functions (Page 845)
ConPerMon modes (Page 845)
Description of ConPerMon (Page 841)

## 6.2.5 ConPerMon messaging

### Messaging

If the control performance falls below a defined limit a message is generated. This is also the case when a defined limit for overshoot is exceeded when there is a step change in the setpoint.

If the CPI temporarily falls below the configured warning and alarm limits, it is not necessary to trigger an alarm immediately. The main aim of the control loop monitoring is to signal the need for maintenance or optimization measures in individual control loops. With the alarm delay, you can make sure that an alarm is triggered only after the cause exists for longer than a configured period `AlmDelay`.

The following messages can be generated for this block:

- Process messages
- Instance-specific messages

**Note**

For ConPerMon, the default value of `RunUpCyc` is set very high (32000). You must adapt it to your own process by taking into account the freely selectable signal inputs `ExtMsg1..3`.

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvID` | SIG 1 | Alarm - high | $$BlockComment$$<br>Overshoot - high alarm limit violated |
| | SIG 2 | Warning - high | $$BlockComment$$<br>Overshoot - high warning limit violated |
| | SIG 3 | Warning - low | $$BlockComment$$<br>`CPI` - low warning limit violated |
| | SIG 4 | Alarm - low | $$BlockComment$$<br>`CPI` - low alarm limit violated |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Instance-specific messages

You can use up to three instance-specific messages with this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvID | SIG 5 | AS process control message - fault | $$BlockComment$$<br>External message 1<br>Status 16#@5%x@ |
| | SIG 6 | AS process control message - fault | $$BlockComment$$<br>External message 2<br>Status 16#@6%x@ |
| | SIG 7 | AS process control message - fault | $$BlockComment$$<br>External message 3<br>Status 16#@7%x@ |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance **MsgEvID**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |
| 4 | CPI (Control Performance Index) |
| 5 | ExtMsg1.ST |
| 6 | ExtMsg2.ST |
| 7 | ExtMsg3.ST |
| 8 | ExtVa108 |
| 9 | ExtVa109 |
| 10 | Reserved |

The associated values 8 ... 9 are allocated to the parameters ExtVa108 ... ExtVa109 and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of ConPerMon (Page 841)

ConPerMon functions (Page 845)

ConPerMon I/Os (Page 860)

ConPerMon block diagram (Page 866)

ConPerMon error handling (Page 857)

ConPerMon modes (Page 845)

## 6.2.6 ConPerMon I/Os

I/Os of ConPerMon

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AlmDelay | Alarm delay time [s] for monitoring the control performance index CPI<br>0 = Alarm delay deactivated | REAL | 0.0 |
| BatchEn | 1 = Enable allocation for batch control | BOOL | 0 |
| BatchID | Current batch ID | DWORD | 16#00000000 |
| BatchName | Current batch designation | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BreakSuppress | 1 = Manually cancel suppression of the control performance alarm during a step response | BOOL | 0 |
| CPI_AlmHyst | Alarm hysteresis of the control performance index [%] | REAL | 5.0 |
| CPI_AL_En | 1 = Activate alarm (low) for monitoring of control performance | BOOL | 1 |
| CPI_AL_Lim | Low alarm limit for control performance [%] | REAL | 30.0 |
| CPI_FiltFactor | Low pass filter for CPI, filter time constant = TimeWindow · CPI_FiltFactor | REAL | 1.0 |
| CPI_WL_En | 1 = Activate warning (low) for monitoring of control performance | BOOL | 1 |
| CPI_WL_Lim | Low warning limit for control performance [%] | REAL | 50.0 |
| CPI_AL_MsgEn | 1 = Activate alarm message for:<br>Low limit for control performance | BOOL | 0 |
| CPI_WL_MsgEn | 1 = Activate warning message for:<br>Low limit for control performance | BOOL | 0 |
| EN | 1 = Called block will be processed | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg3 | Binary input for freely selectable message 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa108 | Associated value 8 for messages (MsgEvID) | ANY | |
| ExtVa109 | Associated value 9 for messages (MsgEvID) | ANY | |
| Feature | I/O for additional functions (Page 845) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| InitRefVar | 1 = Initialization of the block.<br>The benchmark of the controlled variable variance and the reference values of the controlled variable and manipulated variable are measured in the steady state. | BOOL | 0 |
| LoopClosed | 1 = Control loop is closed | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManSupprCPI | 1 = Manual suppression of the CPI calculation and message, e.g. during known disturbances | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgEvID | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_Mon | Manipulated variable of the controller for monitoring | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#78 |
| MV_Ref | Reference value of the manipulated variable | REAL | 0.0 |
| MV_Unit | Unit of measure for manipulated variable | INT | 1342 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OosLi | 1 = Edge transition (0-1) = "Out of service", via interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>0<br>16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 845) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| OS1Perm | I/O for operator permissions (Page 845) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| OvsAH_En | 1 = Activate alarm (high) for overshoot | BOOL | 1 |
| OvsAH_Lim | Overshoot alarm limit [%] | REAL | 25.0 |
| OvsAH_MsgEn | 1 = Enable alarm message for overshoot | BOOL | 0 |
| OvsWH_En | 1 = Activate warning (high) for overshoot | BOOL | 1 |
| OvsWH_Lim | Overshoot warning limit [%] | REAL | 20.0 |
| OvsWH_MsgEn | 1 = Enable warning message for overshoot | BOOL | 0 |
| PV_Mon | Controlled variable for monitoring | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_Ref | Reference value for controlled variable | REAL | 0.0 |
| PV_Unit | Unit of measure for process value | INT | 1001 |
| RefVarExt | Reference value for PV_Variance in control loop "good" status | REAL | 0.0 |
| ReVaExOn | 1 = Use the external reference value of RefVarExt | BOOL | 0 |
| RefVariance | Variance of controlled variable in control loop "good" status | REAL | 100.0 |
| RunUpCyc | Number of start cycles in which messages are suppressed | INT | 32000 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | 1 = Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SelFp2 | 1 = Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | - |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SettlingTimer | Settling time for the adjustment of the dead band | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_Mon | Setpoint of the corresponding controller for monitoring | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| StGainValid | 0 = Output parameter StatGain systematically invalid, e.g. for multi-variable processes | BOOL | 1 |
| TimeWindow | Width of the sliding time window [s] for statistical evaluations | REAL | 120.0 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OSs | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| CPI | Control performance index | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| CPI_AL_Act | 1 = Alarm due to control performance is active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CPI_Suppress | 1 = Message for the control performance index is suppressed; retain last valid CPI value | BOOL | 1 |
| CPI_SuRoot | 1= CPI message suppression was triggered in this control loop | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CPI_WL_Act | 1 = Warning due to control performance is active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ER_Mean | Mean value of the error signal in the time window [PV_Unit] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see ConPerMon error handling (Page 857) | INT | -1 |
| MsgAckn | Message acknowledgment status (output ACK_STATE of ALARM_8P) | WORD | 16#0000 |
| MsgErr | 1 = Message error (output ERROR of ALARM_8P) | BOOL | 0 |
| MsgStatus | Message status (output STATUS of ALARM_8P) | WORD | 16#0000 |
| MV_Mean | Mean value of the manipulated variable in the time window [MV_Unit] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| O_MS_Ext | Reserved | DWORD | 0 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1Perm | DWORD | 16#FFFFFFFF |
| OS1PermLog | Display of OS1Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OverAbso | Absolute overshoot of the step response [PV_Unit] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Overshoot | Relative overshoot of the step response with respect to the step change height [%] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OvsAH_Act | 1 = Alarm due to overshoot is active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OvsWH_Act | 1 = Warning due to overshoot is active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Mean | Mean value of the controlled variable in the time window [PV_Unit] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_StdDev | Standard deviation of the controlled variable | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_ToleHi | Limit (high) of the 3σ band around the setpoint | REAL | 0.0 |
| PV_ToleLo | Limit (low) of the 3σ band around the setpoint | REAL | 0.0 |
| PV_Variance | Variance of the controlled variable | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_VarMin | Minimum observed value of the process variance (slave pointer) | REAL | 10000.0 |
| RefStdDev | Standard deviation of controlled variable in control loop "good" status | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RiseTime | Rise time of the step response [s] | REAL | 0.0 |
| SettliDeadBand | Settling time for the adjustment of the dead band | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SettlRatio | Ratio = Rise time/settling time · 100% | REAL | 0.0 |
| SettliTime | Settling time of the step response [s] | REAL | 0.0 |
| StatGain | Steady-state process gain [PV_Unit / MV_Unit] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 841) | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| `Status2` | Status word 2 (Page 841) | DWORD | 16#00000000 |
| `StepPhase` | Phase of the step response:<br>0 = Waiting<br>1 = Rising<br>2 = Overshoot<br>3 = Settled | INT | 0 |

**See also**

ConPerMon messaging (Page 857)

ConPerMon block diagram (Page 866)

ConPerMon modes (Page 845)

## 6.2.7 ConPerMon block diagram

### ConPerMon block diagram

A block diagram is not provided for this block.

**See also**

ConPerMon I/Os (Page 860)

ConPerMon messaging (Page 857)

ConPerMon error handling (Page 857)

ConPerMon functions (Page 845)

ConPerMon modes (Page 845)

Description of ConPerMon (Page 841)

## 6.2.8 Operator control and monitoring

### 6.2.8.1 ConPerMon views

### Views of the ConPerMon block

The block ConPerMon provides the following views:

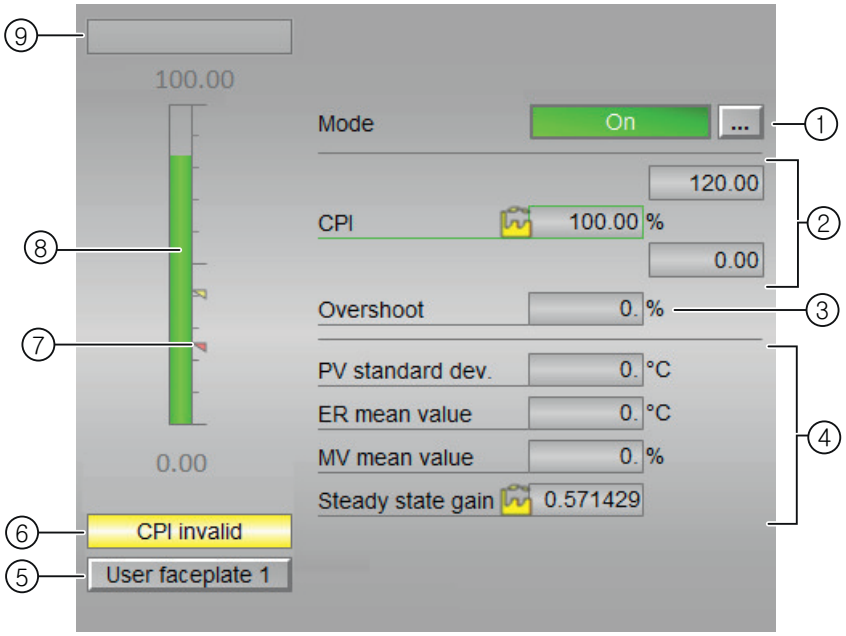
- ConPerMon standard view (Page 867)
- Alarm view (Page 336)

- ConPerMon limit view (Page 869)
- Trend view (Page 341)
- ConPerMon parameter view (Page 870)
- ConPerMon preview (Page 871)
- Memo view (Page 338)
- Batch view (Page 336)
- ConPerMon setpoint view (Page 872)
- Block icon for ConPerMon (Page 874)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 6.2.8.2 ConPerMon standard view

#### ConPerMon standard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 77)
- Out of service (Page 77)

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the operating mode.

### (2) Display area for control performance

This area shows the current control performance index.

### (3) Display area for the overshoot

This area shows you the relative overshoot based on a step change [%].

### (4) Display area for the static evaluation of the current time window (`TimeWindow`)

This area shows you the statistical evaluation of the current time window. The following values are evaluated:

- "PV standard dev.": Standard deviation of the controlled variable
- "ER mean value": Mean value of the control deviation
- "MV mean value": Mean value of the manipulated variable
- "Steady state gain": Steady-state process gain

### (5) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. You can also see the high low limits of the bargraph's scale. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See also the Opening additional faceplates (Page 234) section for more on this.

### (6) Display for CPI valid / CPI invalid

This area shows you if the control performance index is valid or invalid:

- "CPI invalid": Control performance is invalid

You set the limits for the control performance index in the limits views, depending on the configuration in the engineering system (ES).

### (7) Limit display

These colored triangles show you the configured limits in the respective bar graph.

### (8) Bar graph for control performance index

This area shows you the current CPI control performance index in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

For more information, refer: Bar graph (Page 376)

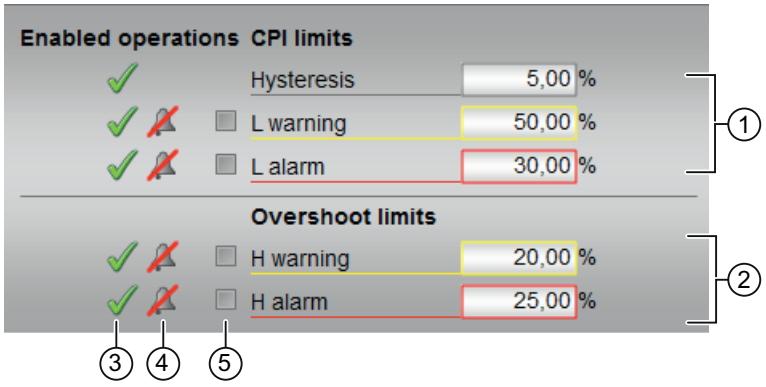
### (9) Display area for block states

This area provides additional information on the operating state of the block:

- Internal error
- Invalid Signal

## 6.2.8.3 ConPerMon limit view

### Limit view of ConPerMon




### (1) CPI limits

In this area, you can enter the limits for the CPI control performance index. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "Hysteresis"
- "L warning": Warning low
- "L alarm": Alarm low

### (2) Overshoot limits

In this area, you can enter the limits for the overshoot. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H warning": Warning high
- "H alarm": Alarm high

### (3) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

### (4) "Message suppression/delay"

Message suppression indicates whether or not the suppression of the associated message in the AS block is activated with the `xx_MsgEn` parameter. The output of messages is not suppressed when the block is installed (all `xx_MsgEn` parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

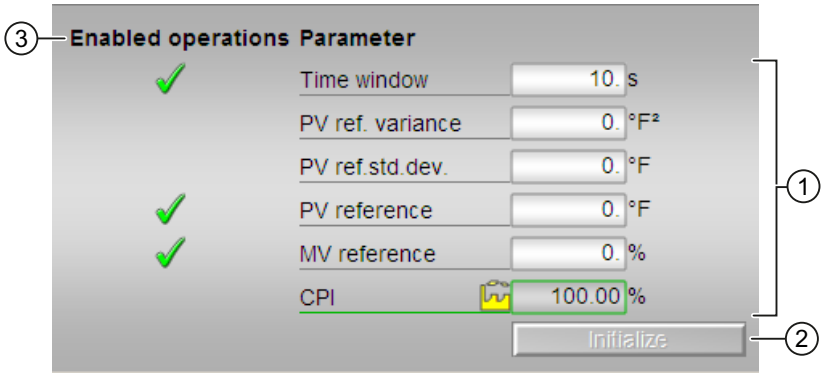
Alarm delays are also displayed in this position; for more on this see section Area of application of the alarm delays (Page 225).

### (5) Suppress messages

You can enable / disable messages by setting the check mark.

## 6.2.8.4 ConPerMon parameter view

### Parameter view of ConPerMon




### (1) Parameter

In this area, you change parameters and therefore influence the controller. Refer to the Changing values (Page 291) section for more on this.

You can influence the following parameters:

- "Time window": Set the time window here, in which the statical evaluation for the following values is to be performed:
  - Standard deviation of the controlled variable
  - Mean value of the control deviation
  - Mean value of the manipulated variable
  - Steady-state process gain
- "PV reference": Reference value for controlled variable
- "MV reference": Reference value of the manipulated variable

### (2) Initialize button

Clicking this button initializes the block. The benchmark of the controlled variable variance and the reference values of the controlled variable and manipulated variable are measured in the steady state.

### (3) Enabled operations

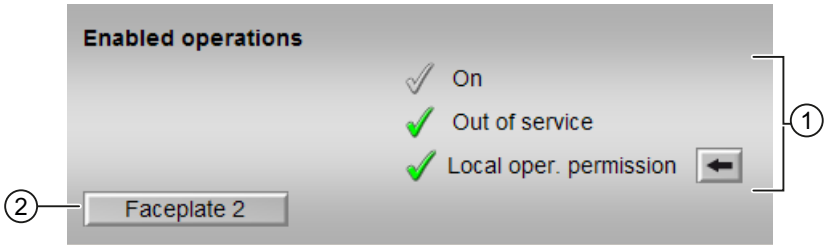
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`)

## 6.2.8.5 ConPerMon preview

### Preview of ConPerMon



### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. You can find more information about this in the section titled: Operator control permissions (Page 285)

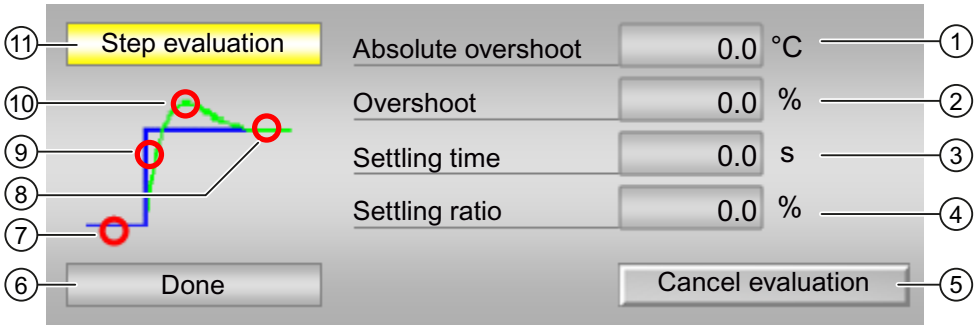
### (2) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

Additional information is available in the section Opening additional faceplates (Page 234).

## 6.2.8.6 ConPerMon setpoint view

### Setpoint view of ConPerMon




### (1) Absolute overshoot

The absolute overshoot is given in the physical unit of the actual value.

### (2) Overshoot

Output of the relative overshoot base on a step change.

### (3) Settling time

Settling time of the step response in seconds.

### (4) Settling time ratio

The settling time ratio is formed from the ramp time by the settling time.

### (5) Cancel evaluation button

You can use the button to show the evaluation of the step response.

**Note**

The "Cancel evaluation" button is operable when **all** of the following conditions are met:

- Operator permission level = 2 (Higher-level process control)
- Parameter `OS_Perm.Bit30` = 1 (Operator can abort the evaluation of the step response)

### (6), (7), (8), (9) and (10): Status of the step response

The following states are shown here:

- (6) Textual display of the states
- (7) "Idle": steady state
- (8) "Steady-state": i.e. the actual value is located within the tolerance band of the setpoint
- (9) "Rising phase": from the initial state to the first time the setpoint is reached
- (10) "Overshoot"

### (11) Display: Evaluation of the step response in progress:

- "Step evaluation"
- "Constant PV"

### See also

ConPerMon standard view (Page 867)
ConPerMon limit view (Page 869)
ConPerMon parameter view (Page 870)

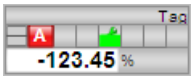
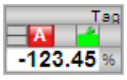
### 6.2.8.7 Block icon for ConPerMon

## Block icons for ConPerMon

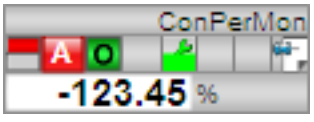
A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits
- Operating modes
- Signal status, release for maintenance
- Memo display
- Process value (black, with and without decimal places)

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | Block icon in the full display |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

# 6.3 FmCont - Interface to module FM 355

## 6.3.1 Description of FmCont

### Object name (type + number) and family

Type + number: FB 1818

Family: Control

### Area of application for FmCont

The block is used for the following applications:

- Fixed setpoint control
- Cascade control
- Ratio control
- Split-range control

### How it works

Block FmCont is used to interface the `FM 355` controller modules.

FmCont can be used for the C (continuous controllers) and S (step and pulse controllers) module types. It contains the algorithms of the setpoint ramp, the setpoint rise limitation, and the limit monitoring of the process value, the control deviation, and the position feedback. Limit monitoring is not used on the module. The control function itself (e.g. PID algorithm) is processed on the module.

You can use the FmCont block to monitor all relevant process values and to change all relevant controller parameters.

Application examples of the FM 355 and detailed descriptions of the associated input and output parameters can be found in the manual for the FM 355. controller module.

Process values such as temperatures, levels and flows can be controlled. However, pressure processes which are not excessively fast are also possible.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100). Set the input parameter `LogAddr` to the module address from HW Config and the input parameter `Channel` to the desired controller channel (1 ... 4).

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The parameter `CoordNo` is set
- The in/out parameter `Mode` is interconnected to the corresponding `OMODE_xx` output parameter of the MOD block.

- The parameter FM 355 is set in accordance with the module type C/S
- The in/out parameter EnCoord is interconnected to the output EN_CO_x of the FM_CO block of the Basis Library (x = number of the rack)
- The output parameter EnCoNum is interconnected to the input ENCOx_yy of the FmCont block (x = number of the rack, yy = coordination number).

For the control loop monitoring to work as planned in the trend view of the controller faceplates, the

S7_xarchive:='Value,shortterm;'

attributes in the process tag types for control loops at the controller function block must be set for the following tags:

- Input parameters:
  - CPI_In
- Output parameters
  - MV
  - MV_HiAct
  - MV_LoAct
  - LoopClosed
  - SP
  - PV_Out
  - PV_ToleHi
  - PV_ToleLo

## Startup characteristics

Use the Feature startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at RunUpCyc.

## Status word allocation for Status1 parameter

For a description of the individual parameters, see the section I/Os of FmCont (Page 897)

| Status bit | Parameter |
|---|---|
| 0 | Occupied |
| 1 | BatchEn |
| 2 | SimOn |
| 3 | OosAct.Value |
| 4 | OosLi.Value |
| 5 | AutAct.Value |
| 6 | Invalid signal status |
| 7 | ManAct.Value |

| Status bit | Parameter |
|---|---|
| 8 | SP_ExtAct.Value |
| 9 | MV_SafeOn.Value AND NOT OosAct.Value |
| 10 | MV_TrkOn.Value AND NOT (ManAct.Value OR OosAct.Value OR MV_SafeAct.Value) |
| 11 | MV.Value > ManLoLim for continuous or pulse controller<br>NOT FbkClosed.Value for step controller with/without position feedback |
| 12 | Open.Value |
| 13 | Close.Value |
| 14 | Stop.Value |
| 15 | FbkOpnOut.Value |
| 16 | FbkClsOut.Value |
| 17 | SimOn AND ManAct |
| 18 | MV_SafeOn.Value |
| 19 | AdvCoAct.Value |
| 20 | 1 = Input parameter Rbk is not interconnected (RbkOut.ST = 16#FF) |
| 21 | NegGain.Value |
| 22 | MV_FmTrkAct.Value AND NOT (OosAct.Value OR MV_SafeAct.Value) |
| 23 - 24 | Not used |
| 25 | MV_TrkOn.Value |
| 26 | MV_FmTrkOn.Value |
| 27 | AdvCoModSP |
| 28 | 1 = Analog controller (FM 355 = 1) |
| 29 | 1 = Pulse controller (FM 355 = 0 AND StepCon = 0) |
| 30 | 1 = Step controller with position feedback<br>(FM 355 = 0 AND StepCon = 1 AND WithRbk = 1) |
| 31 | 1=Step controller without position feedback<br>(FM 355 = 0 AND StepCon = 1 AND WithRbk = 0) |

## Status word allocation for Status2 parameter

| Status bit | Parameter |
|---|---|
| 0 | MsgLock |
| 1 | PV_AH_Act.Value |
| 2 | PV_WH_Act.Value |
| 3 | PV_TH_Act.Value |
| 4 | PV_TL_Act.Value |
| 5 | PV_WL_Act.Value |
| 6 | PV_AL_Act.Value |
| 7 | PV_AH_En OR Feature.Bit31 |
| 8 | PV_WH_En OR Feature.Bit31 |
| 9 | PV_TH_En OR Feature.Bit31 |
| 10 | PV_TL_En OR Feature.Bit31 |
| 11 | PV_WL_En OR Feature.Bit31 |

| Status bit | Parameter |
|---|---|
| 12 | PV_AL_En OR Feature.Bit31 |
| 13 | PV_AH_MsgEn |
| 14 | PV_WH_MsgEn |
| 15 | PV_TH_MsgEn |
| 16 | PV_TL_MsgEn |
| 17 | PV_WL_MsgEn |
| 18 | PV_AL_MsgEn |
| 19 | ER_AH_Act.Value |
| 20 | ER_AL_Act.Value |
| 21 | ER_AH_En OR Feature.Bit31 |
| 22 | ER_AL_En OR Feature.Bit31 |
| 23 | ER_AH_MsgEn |
| 24 | ER_AL_MsgEn |
| 25 | RbkWH_Act.Value |
| 26 | RbkWL_Act.Value |
| 27 | RbkWH_En OR Feature.Bit31 |
| 28 | RbkWL_En OR Feature.Bit31 |
| 29 | RbkWH_MsgEn |
| 30 | RbkWL_MsgEn |
| 31 | MS_RelOp |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 | Effective signal 1 of the message block connected via EventTsIn |
| 1 | Effective signal 2 of the message block connected via EventTsIn |
| 2 | Effective signal 3 of the message block connected via EventTsIn |
| 3 | Effective signal 4 of the message block connected via EventTsIn |
| 4 | Effective signal 5 of the message block connected via EventTsIn |
| 5 | Effective signal 6 of the message block connected via EventTsIn |
| 6 | Effective signal 7 of the message block connected via EventTsIn |
| 7 | Effective signal 8 of the message block connected via EventTsIn |
| 8 | Delay of the PV_AH_Lim message |
| 9 | Delay of the PV_WH_Lim message |
| 10 | Delay of the PV_TH_Lim message |
| 11 | Delay of the PV_TL_Lim message |
| 12 | Delay of the PV_WL_Lim message |
| 13 | Delay of the PV_AL_Lim message |
| 14 | Delay of the ER_AH_Lim message |
| 15 | Delay of the ER_AL_Lim message |
| 16 | Collection of message delays |
| 17 - 19 | Not used |

| Status bit | Parameter |
|---|---|
| 20 | 1= Suppressed messages |
| 21 - 23 | Not used |
| 24 | Internal error |
| 25-26 | Not used |
| 27 | `SP_UpRaAct`, `SP_DnRaAct` limits enabled for gradient mode (`SP_RateOn = 1`) |
| 28 | `GrpErr.Value` |
| 29 | `RdyToStart.Value` |
| 30 | `SimLiOp.Value` |
| 31 | Not used |

### Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 - 7 | Effective signal 9..16 of the message block connected via `EventTsIn` |
| 8 - 31 | Not used |

### See also

FmCont messaging (Page 894)

FmCont block diagram (Page 913)

FmCont modes (Page 880)

FmCont error handling (Page 892)

FmCont functions (Page 882)

## 6.3.2 FmCont modes

### FmCont operating modes

The block can be operated using the following modes:

- Automatic mode (Page 79)
- Manual mode (Page 79)
- Program mode for controllers (Page 85)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and bumpless switchover in the Manual and automatic mode for control blocks (Page 79) section.

### "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the Manual and automatic mode for control blocks (Page 79) section.

### "Program mode for controllers"

General information on "Program mode for controllers" is available in the section Program mode for controllers (Page 85).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit5` =1 and `Feature.Bit5` =0 is as shown below:

| Output Parameters | **`Feature.Bit5` =0** | **`Feature.Bit5` =1** |
|---|---|---|
| RbkOut.ST | 16#XX | 16#60 |
| RbkWH_Act.ST | 16#XX | 16#60 |
| RbkWL_Act.ST | 16#XX | 16#60 |
| FbkOpnOut.ST | 16#80 | 16#60 |
| FbkClsOut.ST | 16#80 | 16#60 |
| SP_ExtAct.ST | 16#80 | 16#60 |
| SP.ST | 16#80 | 16#60 |
| SP_ExtOut.ST | 16#80 | 16#60 |
| PV.ST | 16#80 | 16#60 |
| PV_Out.ST | 16#80 | 16#60 |
| PV_AH_Act.ST | 16#XX | 16#60 |
| PV_TH_Act.ST | 16#XX | 16#60 |
| PV_TL_Act.ST | 16#XX | 16#60 |
| PV_WL_Act.ST | 16#XX | 16#60 |
| PV_AL_Act.ST | 16#XX | 16#60 |
| PV_ToleHi.ST | 16#XX | 16#60 |
| PV_ToleLo.ST | 16#XX | 16#60 |
| ER.ST | 16#80 | 16#60 |
| ER_AH_Act.ST | 16#XX | 16#60 |
| ER_AL_Act.ST | 16#XX | 16#60 |

### See also

FmCont block diagram (Page 913)
FmCont I/Os (Page 897)
Description of FmCont (Page 876)
FmCont functions (Page 882)
FmCont error handling (Page 892)
FmCont messaging (Page 894)

## 6.3.3 FmCont functions

### Functions of FmCont

The functions for this block are listed below.

### Module types

FmCont can be used for the C (continuous controllers) and S (step controllers with and without position feedback and pulse controllers) module types. You can use the following parameters to identify which module type and controller type has been set:

| FM 355 | StepCon | WithRbk | Module type, controller type |
|---|---|---|---|
| 1 or C | - | - | FM 355 C: Continuous controller |
| 0 or S | 1 | 1 | FM 355 S: Step controller with position feedback |
| 0 or S | 1 | 0 | FM 355 S: Step controller without position feedback |
| 0 or S | 0 | - | FM 355 S: Pulse controller |

You must set input parameter `StepCon` if you want to set the step controller with/without position feedback as the controller type.

### Generating manipulated variables for continuous controllers, step controllers with position feedback, or pulse controllers

The manipulated variable MV and the actuating signals Open, Close and Stop are generated as follows

| MV_Safe-On | MV_FmTrk On | ManAct | MV_TrkOn | AdvCoAct AND NOT AdvCo-ModSP | MV = | Manipu-lated varia-ble limit | State | Open, Close, Stop |
|---|---|---|---|---|---|---|---|---|
| 1 | - | - | - | - | MV_Safe | MV_HiLim<br>MV_LoLim | Tracking to safety value | **Cont. controller:** Open, Close, Stop = 0<br>**Step controller with position feedback:** Depending on Rbk and MV, the output signals Open, Close and Stop are generated using the algorithm of a positioner.<br>**Pulse controller:** Depending on MV, the output signals Open and Close are generated using the algorithm of a pulse controller (Stop = 0). |
| 0 | 1 | - | - | - | Prepared FM analog input | MV_HiLim<br>MV_LoLim | Tracking to an FM analog input | |
| 0 | 0 | 1 | - | - | Man | ManHiLim<br>ManLoLim | Manual mode, set by the operator | |
| 0 | 0 | 0 | 1 | - | MV_Trk | MV_HiLim<br>MV_LoLim | Tracking to block input MV_Trk | |
| 0 | 0 | 0 | 0 | 1 | AdvCoMV | MV_HiLim<br>MV_LoLim | Higher-level program mode | |
| 0 | 0 | 0 | 0 | 0 | P_Part + I_Part + D_Part + FFwd | MV_HiLim<br>MV_LoLim | Automatic mode (PID algorithm) | |

### Generating actuating signals for step controllers without position feedback (WithRbk = 0)

The manipulated variable signals `Open`, `Close` and `Stop` can be generated as follows:

| ManAct | Open, Close, Stop | State |
|---|---|---|
| 1 | The output signals are generated using input signals `OpenOp`/`Li`, `CloseOp`/`Li` or `StopOp`/`Li` | Manual mode, set by the operator |
| 0 | The output signals are generated using PID output variables `P_Part`, `I_Part`, `D_Part` and `FFwd` | Automatic mode (PID algorithm) |

### Tracking and limiting a manipulated variable (cont. controller, step controller with position feedback and pulse controller)

The block provides the standard function Tracking and limiting a manipulated variable (Page 222).

### Neutral position

The controller modules have their own mechanism for feedforwarding a safety value (see manual for Temperature Controller FM 355-2 or manual for Controller Module FM 355).

### Group error

This block provides the standard function Outputting group errors (Page 130).

The following parameters are taken into consideration when forming the group error:

- `CSF`
- `ModErr`
- `ParFM_Err`
- `PerACCErr`

### Outputting a signal for start readiness

This block provides the standard function Outputting a signal for start readiness (Page 59).

### "Actuator active" information

For continuous and pulse controllers: If the manipulated variable `MV` is greater than the minimum manual limit `ManLoLim`, this is recognized as actuator active.

For step controllers: If the parameter `FbkClosed` = 0 , this is recognized as "Actuator active".

This status can be used to indicate a customized symbol in the process image, for example, and is saved in the status word (see Status word section in Description of FmCont (Page 876)).

### Limit monitoring of position feedback (cont. controller, step controller with position feedback and pulse controller)

The block provides the standard function Limit monitoring of the feedback (Page 101).

### Group display SumMsgAct for limit monitoring, CSF and ExtMsgx

The block provides the standard function Group display for limit monitoring, CSF and ExtMsgx (Page 92).

### External/internal setpoint specification

The block provides the standard function Setpoint specification - internal/external (Page 136).

### Setpoint limiting for external setpoints

The block provides the standard function Setpoint limiting for external setpoints (Page 222).

### Gradient limit of the setpoint

The block provides the standard function Gradient limit of the setpoint (Page 132).

### Using setpoint ramp

The block provides the standard function Using setpoint ramp (Page 131).

### Tracking setpoint in manual mode

The block provides the standard function Tracking setpoint in manual mode (Page 222).

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Process value (`SimPV`, `SimPV_Li`)
- Position feedback (`SimRbk`, `SimRbkLi`)

**Note**

The simulated process value `SimPV` only affects alarm processing and not the PID algorithm in the control module.

### Limit monitoring of the process value

The block provides the standard function Limit monitoring of the process value (Page 93).

### Error signal generation and dead band

The block provides the standard function Error signal generation and dead band (Page 217).

### Limit monitoring of error signal

The block provides the standard function Limit monitoring of setpoint, manipulated variable and control deviation (Page 102).

### Inverting control direction

The block provides the standard function Inverting control direction (Page 217).

### Physical standardization of setpoint, manipulated variable and process value

Controller gain `Gain` is entered either using a physical variable or as standardized value.

- `Gain` as a physical variable:
  The standardized variables retain their default values:
  - `NormPV.High` = 100 and `NormPV.Low` = 0
  - `NormMV.High` = 100 and `NormMV.Low` = 0

  For step controllers with/without position feedback and pulse controllers, the values of `NormMV.High` and `NormMV.Low` are not taken into account. The algorithm uses default values 0 and 100 for internal calculations.
  The effective gain is: `GainEff` = `Gain`
- Entering a standardized `Gain` (dimensionless):
  Change the standardized variables to the actual range of the process values and manipulated variables.
  - Internal and external setpoints; the process value and corresponding parameters are entered according to the physical measuring range of the process value.

  Continuous controller, pulse controller:
  - The manual value, the tracking value of the manipulated variable, feedforward control and the corresponding parameters are set according to the physical measuring range of the manipulated variable.

  Step controller with position feedback:
  - The manual parameter, the tracking value of the manipulated variable, feedforward control and the corresponding parameters are entered as a percentage 0 ... 100.

  Step controller without position feedback:
  - No physical measuring range available.

  The effective gain is:
  - Step controller with/without position feedback:
    `GainEff` = 100.0 / (`NormPV.High` - `NormPV.Low`) · `Gain`
  - Continuous controller, pulse controller:
    `GainEff` = (`NormMV.High` - `NormMV.Low`) / (`NormPV.High` - `NormPV.Low`) · `Gain`

### Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 240).

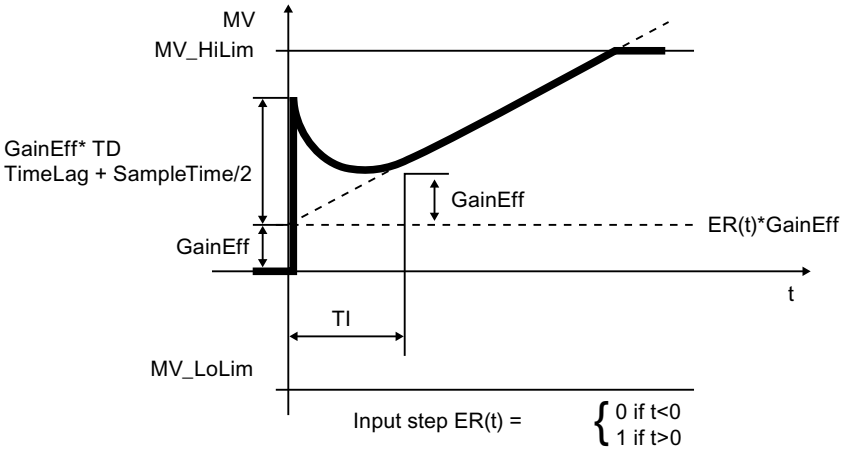
## PID algorithm

The manipulated variable is generated in automatic mode according to the following algorithm:

MV = GainEff · (1 + 1 / (TI · s) + (TD · s) / (1 + TD / DiffGain · s)) · ER

Where: s = Complex number

The following step response occurs:




**Note**

This formula describes a standard application where the P, I and D components are activated and the P and D components are not in the feedback circuit (PropSel = 1, TI ≠ 0, D_InSel = 0 and P_FbkSel = 0).

The D component delay is derived from TD / DiffGain.

- The P component can be shut down by PropSel = 0 .
- The I component can be shut down by TI = 0 .
- The D component can be shut down by TD = 0 .

## Structure segmentation at controllers

The PID controller algorithm of FM 355 features structure segmentation. It is activated via the P_FbkSel and D_InSel parameters. The precise functionality is described in the FM 355 manual.

## Anti-windup

The PID control algorithm of FM 355 has an anti-windup function. The I component is frozen or tracked after the manipulated variable has reached its limits (MV_HiLim or MV_LoLim).

## Feedforwarding and limiting disturbance variables

The block provides a function for activating the disturbance variable feedforward. The precise functionality is described in the FM 355 manual.

### Forming the signal status for blocks

The block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

- Signal status for the setpoint value SP:
  The signal status of the SP output parameter is always equivalent to the signal status of input parameter SP_Ext or SP_Int, depending on how the setpoint is specified. If the internal setpoint SP_Int is used, the signal status is always output as 16#80.
- Signal status for PV_Out, RbkOut, Open, Close, Stop:
  The signal status is always 16#60 when simulation is activated.
  When the ModErr.Value, ChFM_Err, ParFM_Err module error occurs, the signal status of PV_Out is always 16#0. RbkOut is always 16#0 for step controllers with position feedback
  Otherwise, the following applies:
  ```
  PV_Out.ST: 16#80
  Step controller: RbkOut.ST: = 16#80
  Continuous controller or pulse controller: RbkOut.ST: = Rbk.ST
  Open.ST := 16#80;
  Close.ST := 16#80;
  Stop.ST := 16#80;
  ```
- Signal status of the error signal ER:
  The signal status of output parameter ER is obtained from the worst signal status of the two output parameters PV_Out and SP and is output. The signal status 16#60 (external simulation) is suppressed because the block acts as a sink with external simulation.
  Signal status for FbkOpnOut, FbkClsOut:
  ```
  FbkOpnOut.ST := FbkOpened.ST;
  FbkClsOut.ST := FbkClosed.ST;
  ```
- Signal status for the manipulated variable MV:
  The status signal from the output parameter MV is always set to 16#80 in "manual mode" and for step controllers without position feedback.
  In "automatic mode", the signal status for continuous controllers or pulse controllers is formed from the following parameters:
  RbkOut.STFFwdOut.STER.STWith step controllers, the FbkOpnOut.STFbkClsOut.STparameters are also included. The signal status 16#60 (external simulation) is suppressed because the block acts as a sink with external simulation.
- Worst signal status:
  The worst signal status ST_Worst for the block is formed from:
  - PV_Out.ST;
  - SP:ST;
  - FFwdOut.ST;
  - RbkOut.ST;

  With step controllers (FM355 = 0, StepCon = 1), the following are also included:
  - FbkOpnOut.ST;
  - FbkClsOut.ST;

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 4 | Setting switch or button mode (Page 177) |
| 5 | Changes signal status of outputs in OOS (Page 208) |
| 6 | Ramp rate calculation (Page 191) |
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 23 | SP following PV in open loop has no priority over SP_Ext and SP limits (Page 193) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 28 | Disabling operating points (Page 156) |
| 29 | Signaling limit violation (Page 181) |
| 31 | Limits disabled in faceplate (Page 206) |

### Configurable reactions using the `Feature2` parameter

| Bit | Function |
|---|---|
| 11 | Suppress MsgLock and “Out of service” mode for a connected message block (Page 202) |
| 12 | Disable limit act output when limit value message is disabled (Page 209) |

### Operator permissions

The block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" `AutModOp` |
| 1 | 1 = Operator can switch to "manual mode" `ManModOp` |
| 2 | 1 = Operator can switch to "Out of service" mode `OosOp` |
| 3 | 1 = Operator can switch to "program mode" `AdvCoEn` |
| 4 | 1 = Operator can switch the setpoint to "external" `SP_ExtOp` |
| 5 | 1 = Operator can switch the setpoint to "internal" `SP_IntOp` |
| 6 | 1 = Operator can change the internal setpoint `SP_Int` |

| Bit | Function |
|---|---|
| 7 | Continuous controllers, pulse controllers or step controllers with position feedback:<br>1 = Operator can change the manual parameter `Man`<br>Step controller without position feedback:<br>1 = Operator can change the manual operation signals `OpenOp`, `StopOp`, `CloseOp` |
| 8 | 1 = Operator can change operation high limit of the setpoint `SP_InHiLim` |
| 9 | 1 = Operator can change operation low limit of the setpoint `SP_InLoLim` |
| 10 | 1 = Operator can change the operation high limit of the manipulated variable `ManHiLim` |
| 11 | 1 = Operator can change the operation low limit of the manipulated variable `ManLoLim` |
| 12 | 1 = Operator can enable the setpoint's gradient limitation function `SP_RateOn` |
| 13 | 1 = Operator can change the setpoint's high limit for the ramp `SP_UpRaLim` |
| 14 | 1 = Operator can change the setpoint's low limit for the ramp `SP_DnRaLim` |
| 15 | 1 = Operator can switch between the time value or the value for the ramp `SP_RmpModTime` |
| 16 | 1 = Operator can change the ramp time `SP_RmpTime` |
| 17 | 1 = Operator can change the target setpoint `SP_RmpTarget` for the setpoint ramp |
| 18 | 1 = Operator can enable the setpoint ramp function `SP_RmpOn` |
| 19 | Not used |
| 20 | 1 = Operator can enable the track setpoint in manual mode function `SP_TrkPV` |
| 21 | 1 = Operator can enable the bumpless switchover from external to internal `SP_TrkExt` |
| 22 | 1 = Operator can change the gain parameter `Gain` |
| 23 | 1 = Operator can change the integral time parameter `TI` |
| 24 | 1 = Operator can change the derivative time parameter `TD` |
| 25 | 1 = Operator can change the derivative gain parameter `DiffGain` |
| 26 | 1 = Operator can change the dead band parameter `DeadBand` |
| 27 | Not used |
| 28 | 1 = Operator can change the integral time parameter `MotorTime` |
| 29 | 1 = Operator can change the integral time parameter `PulseTime` |
| 30 | 1 = Operator can change the integral time parameter `BreakTime` |
| 31 | Not used |

The block has the following permissions for the `OS1Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can change the limit (process value) `PV_AH_Lim` for the high alarm |
| 1 | 1 = Operator can change the limit (process value) `PV_WH_Lim` for the high warning |
| 2 | 1 = Operator can change the limit (process value) `PV_TH_Lim` for the high tolerance |
| 3 | 1 = Operator can change the hysteresis (process value) `PV_Hyst` |
| 4 | 1 = Operator can change the limit (process value) `PV_TL_Lim` for the low tolerance |
| 5 | 1 = Operator can change the limit (process value) `PV_WL_Lim` for the low warning |
| 6 | 1 = Operator can change the limit (process value) `PV_AL_Lim` for the low alarm |
| 7 | 1 = Operator can change the limit (error signal) `ER_AH_Lim` for the high alarm |
| 8 | 1 = Operator can change the hysteresis (error signal) `ER_Hyst` |
| 9 | 1 = Operator can change the limit (error signal) `ER_AL_Lim` for the low alarm |
| 10 | 1 = Operator can change the limit (position feedback) `RbkWH_Lim` for the high warning |
| 11 | 1 = Operator can change the hysteresis (position feedback) `RbkHyst` |

| Bit | Function |
|---|---|
| 12 | 1 = Operator can change the limit (position feedback) RbkWL_Lim for the low warning |
| 13 | 1 = Operator can open the valve |
| 14 | 1 = Operator can close the valve |
| 15 | 1 = Operator can stop the valve |
| 16 | 1 = Operator can activate the Simulation function SimOn |
| 17 | 1 = Operator can activate the Release for maintenance function MS_RelOp |
| 18 | 1 = Operator can activate / deactivate messages via PV_AH_MsgEn |
| 19 | 1 = Operator can activate / deactivate messages via PV_WH_MsgEn |
| 20 | 1 = Operator can activate / deactivate messages via PV_TH_MsgEn |
| 21 | 1 = Operator can activate / deactivate messages via PV_TL_MsgEn |
| 22 | 1 = Operator can activate / deactivate messages via PV_WL_MsgEn |
| 23 | 1 = Operator can activate / deactivate messages via PV_AL_MsgEn |
| 24 | 1 = Operator can activate / deactivate messages via ER_AH_MsgEn |
| 25 | 1 = Operator can activate / deactivate messages via ER_AL_MsgEn |
| 26 | 1 = Operator can activate / deactivate messages via RbkWH_MsgEn |
| 27 | 1 = Operator can activate / deactivate messages via RbkWL_MsgEn |
| 28 | 1 = Operator can change the simulation value SimPV |
| 29 | 1 = Operator can change the simulation value SimRbk |
| 30 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in OS_Perm as a parameter, you have to reset the corresponding OS_Perm bit.

### Release for maintenance

The block provides the standard function Release for maintenance (Page 70).

### Generating instance-specific messages

The block provides the standard function Generating instance-specific messages (Page 230) without the time stamp function in the I/O.

### Suppressing messages using the MsgLock parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Specifying the display area for process and setpoint values as well as operations

This block provides the standard function Display and operator input area for process values and setpoints (Page 233).

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Button labels

This block provides the standard function Labeling of buttons and text (Page 235)

Instance-specific text can be configured for the following parameters:

- `OpenOp`
- `StopOp`
- `CloseOp`

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### See also

FmCont messaging (Page 894)

FmCont I/Os (Page 897)

FmCont block diagram (Page 913)

FmCont modes (Page 880)

FmCont error handling (Page 892)

EventTs functions (Page 2046)

## 6.3.4 FmCont error handling

### Error handling of FmCont

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Control system fault (CSF)

## Overview of error numbers

The ErrorNum I/O can be used to output various error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. |
| 30 | The value of PV can no longer be displayed in the REAL number field. |
| 31 | The value of SP_Ext can no longer be displayed in the REAL number field. |
| 33 | The value of MV_Trk can no longer be displayed in the REAL number field. |
| 35 | The value of Rbk can no longer be displayed in the REAL number field. |
| 36 | The value of MV can no longer be displayed in the REAL number field. |
| 50 | The controller cannot be switched to program mode, because program mode with default setpoint (AdvCoModSP = 0) is not possible for step controllers without position feedback (WithRbk = 0). |
| 51 | AutModLi = 1 and ManModLi = 1<br>SP_LiOp = 1 and SP_IntLi = 1 and SP_ExtLi = 1<br>OpenLi = 1 and StopLi = 1<br>CloseLi = 1 and StopLi = 1<br>OpenLi = 1 and CloseLi = 1 |
| 59 | = 1, "Gain is negative" |
| 60 | \|TI\| < SampleTime / 2 |
| 61 | \|TD\| < SampleTime |
| 62 | DiffGain < 1 or DiffGain > 10 |
| 63 | TD / DiffGain < SampleTime / 2 |
| 64 | PropFacSP < 0 or PropFacSP > 1 |
| 66 | NormPV_High = NormPV_Low |
| 67 | MotorTime < SampleTime |
| 68 | PulseTime < SampleTime |
| 69 | BreakTime < SampleTime |
| 70 | Channel < 1 or Channel > 4 |
| 71 | (D_InSel < 0 or D_InSel > 4) and D_InSel ≠ 17 |

## Control system fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 128) section for more on this.

## See also

FmCont block diagram (Page 913)

FmCont I/Os (Page 897)

Description of FmCont (Page 876)

FmCont modes (Page 880)

FmCont functions (Page 882)

FmCont messaging (Page 894)
Setting switch or button mode (Page 177)

## 6.3.5 FmCont messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Process messages
- Instance-specific messages

### Process control fault

The following control system error messages can be output:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId2` | SIG 6 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter `CSF`. If it changes to `CSF` = 1, a process control fault is triggered (`MsgEvId2`, SIG 6).

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 1 | Alarm - high | $$BlockComment$$<br>PV - high alarm limit violated |
| | SIG 2 | Warning - high | $$BlockComment$$<br>PV - high warning limit violated |
| | SIG 3 | Tolerance - high | $$BlockComment$$<br>PV - high tolerance limit violated |
| | SIG 4 | Tolerance - low | $$BlockComment$$<br>PV - low tolerance limit violated |
| | SIG 5 | Warning - low | $$BlockComment$$<br>PV - low warning limit violated |
| | SIG 6 | Alarm - low | $$BlockComment$$<br>PV - low alarm limit violated |
| | SIG 7 | Alarm - high | $$BlockComment$$<br>ER - high alarm limit violated |
| | SIG 8 | Alarm - low | $$BlockComment$$<br>ER - low alarm limit violated |
| MsgEvId2 | SIG 7 | Warning - high | $$BlockComment$$<br>Rbk - high warning limit violated |
| | SIG 8 | Warning - low | $$BlockComment$$<br>Rbk - low warning limit violated |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Instance-specific messages

You can use up to four instance-specific messages with this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId2 | SIG 1 | AS process control message - fault | $$BlockComment$$<br>External message 1<br>Status 16#@5%x@ |
| | SIG 2 | AS process control message - fault | $$BlockComment$$<br>External message 2<br>Status 16#@6%x@ |
| | SIG 3 | AS process control message - fault | $$BlockComment$$<br>External message 3<br>Status 16#@7%x@ |
| | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External message 4<br>Status 16#@8%x@ |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance `MsgEvId1`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | Process value `PV_Out` |
| 5 | Control deviation `ER` |
| 6 | `ExtVa106` |
| 7 | `ExtVa107` |
| 8 | Not allocated |
| 9 | Not allocated |
| 10 | Not allocated |

The associated values 6 ... 7 are allocated to the parameters `ExtVa106` ... `ExtVa107` and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### Associated values for message instance `MsgEvId2`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | Position feedback `RbkOut` |
| 5 | Signal status `ExtMsg1` |
| 6 | Signal status `ExtMsg2` |
| 7 | Signal status `ExtMsg3` |
| 8 | Signal status `ExtMsg4` |
| 9 | `ExtVa209` |
| 10 | `ExtVa210` |

The associated values 9 ... 10 are allocated to the parameters `ExtVa209` ... `ExtVa210` and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

FmCont block diagram (Page 913)

FmCont modes (Page 880)

FmCont error handling (Page 892)

### 6.3.6 FmCont I/Os

I/Os of FmCont

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

#### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AccMode* | 1 = Transfer of operating parameters SubN1_ID, SubN2_ID, RackNo, SlotNo and Channel to internal processing | BOOL | 1 |
| AdvCoEn | 1 = Enable "program mode" via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AdvCoModSP | Type of "program mode":<br>1 = Setpoint specification<br>0 = Manipulated variable specification | BOOL | 1 |
| AdvCoMstrOn | Activate (0-1) or deactivate (1-0) "program mode" via edge transition | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AdvCoMV | Specified value from the external program.<br>The parameter is available on the WinCC OPC server to be read by external APC programs. | REAL | 0.0 |
| AdvCoOn* | 1 = Enable "program mode" via faceplate.<br>The parameter is available on the WinCC OPC server to be read by external APC programs. | BOOL | 0 |
| AutModLi* | 1 = "Automatic mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutModOp* | 1 = "Automatic mode" via operator (controlled by ModLiOp = 0) | BOOL | 0 |
| BatchEn | 1 = Enable allocation for batch control | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BreakTime* | Minimum break duration [s] | REAL | 1.0 |
| Channel | Controller channel number (1..4) | INT | 1 |
| CloseLi* | 1 = Close via interconnection or CFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CloseOp* | 1 = Close via operator | BOOL | 0 |
| CoordNo | Coordination number | INT | 0 |
| CPI_In | Input for control performance index, which is calculated by the assigned ConPerMon block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#78 |
| CSF | 1 = External error (control system fault) Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| D_InSel* | Input for differentiator:<br>0 = Error signal<br>1..4 = Channel 1..4<br>17 = Actual value to feedback | INT | 0 |
| DeadBand* | Width of dead band | REAL | 0.0 |
| DiffGain* | Gain of differentiator [1...10] DiffGain = TD/(delay time of D component) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 5.0<br>• 16#80 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| ER_A_DC* | Delay for incoming alarms during error signal monitoring | REAL | 0.0 |
| ER_A_DG* | Delay for outgoing alarms during error signal monitoring | REAL | 0.0 |
| ER_AH_En | 1 = Activate alarm (high) for error signal monitoring | BOOL | 1 |
| ER_AH_Lim | Alarm limit (high) for error signal monitoring | REAL | 100.0 |
| ER_AH_MsgEn | 1 = Activate messages for alarm (high) for error signal monitoring | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ER_AL_En | 1 = Activate alarm (low) for error signal monitoring | BOOL | 1 |
| ER_AL_Lim | Alarm limit (low) for error signal monitoring | REAL | -100.0 |
| ER_AL_MsgEn | 1 = Activate messages for alarm (low) for error signal monitoring | BOOL | 1 |
| ER_Hyst | Alarm hysteresis for error signal | REAL | 1.0 |
| EventTsIn | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg3 | Binary input for freely selectable message 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg4 | Binary input for freely selectable message 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| ExtVa209 | Associated value 9 for messages (MsgEvID2) | ANY | |
| ExtVa210 | Associated value 10 for messages (MsgEvID2) | ANY | |
| FbkClosed | Low limit stop signal of position feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>0<br>16#80 |
| FbkOpened | High limit stop signal of position feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Feature | I/O for additional functions (Page 882) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| FM355 | Module type:<br>0: FM 355 S; 1: FM 355 C | BOOL | 0 |
| FuzOptOn* | Fuzzy optimization | BOOL | 0 |
| Gain* | Proportional gain<br>Gain.ST = 16#FF: Enabled in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#FF |
| LogAddr | Logical address FM 355 | INT | 0 |
| Man* | Manual specification for the manipulated variable | REAL | 0.0 |
| ManHiLim* | Limit (high) for manual parameter Man | REAL | 100.0 |
| ManLoLim* | Limit (low) for manual parameter Man | REAL | 0.0 |
| ManModLi* | 1 = "Manual mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManModOp* | 1 = "Manual mode" via OS operator (controlled by ModLiOp = 0) | BOOL | 1 |
| Mode | Operating mode | DWORD | 16#00000000 |
| ModLiOp | Operating mode switchover between:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MotorTime* | Motor actuating time [s] | REAL | 30.0 |
| MS | Maintenance status | DWORD | 0 |
| MS_Release | Release for maintenance (interconnected with MS_Release of the technology block) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ms_Ext | External maintenance status. For more information, refer to the MS_Ext (Page 2463) parameter section under information on using channel blocks. | DWORD | 0 |
| MS_RelOp* | 1 = Release for maintenance by OS operator | BOOL | 0 |
| MsgEvID1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgEvID2 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MV_FmTrkOn | 1 = Manipulated variable tracking in the FM | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_HiLim* | Limit (high) for manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| MV_LoLim* | Limit (low) for manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_OpScale | OS display range for manipulated variable MV | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| MV_Safe* | Neutral position manipulated variable | REAL | 0.0 |
| MV_SafeOn | 1 = Neutral position manipulated variable MV_Safe at output MV | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_Trk* | Tracking value for the manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_TrkOn | 1 = Tracking of manipulated variable MV | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_Unit | Unit of measure for manipulated variable | INT | 1342 |
| NegGain* | 0 = Effective proportional gain GainEff is positive<br>1 = Effective proportional gain GainEff is negative | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NormMV | Manipulated variable range (MV) for standardizing the proportional gain (GAIN) | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| NormPV | Process value range (PV) for standardizing the proportional gain (GAIN) | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| Occupied | Occupied by batch control | BOOL | 0 |
| OosLi | Edge transition (0-1) = "Out of service", via interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OP_Sel* | Operation via OP<br>0 = "Off" (P bus)<br>1 = "On" (K bus) | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OpenLi* | 1 = Open via interconnection or CFC | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| OpenOp* | 1 = Open via operator | BOOL | 0 |
| OptimEn | 1 = Enable optimization of PID parameters by PID tuner | BOOL | 0 |
| OptimOcc | 1 = Optimization running | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 882) | STRUCT | - |
| | | • Bit 0: BOOL | • 1 |
| | | • ... | • 1 |
| | | • Bit 31: BOOL | • 1 |
| OS1Perm | I/O for operator permissions (Page 882) | STRUCT | - |
| | | • Bit 0: BOOL | • 1 |
| | | • Bit 18: BOOL | • 1 |
| | | • Bit 19: BOOL | • 1 |
| | | • Bit 31: BOOL | • 1 |
| P_FbkSel* | 1 = P component in feedback | BOOL | 0 |
| PropSel* | 1 = Activate P component | BOOL | 1 |
| PulseTime* | Minimum pulse duration [s] | REAL | 1.0 |
| PV_A_DC* | Delay time for incoming PV alarms [s] | REAL | 0.0 |
| PV_A_DG* | Delay time for outgoing PV alarms [s] | REAL | 0.0 |
| PV_AH_En | 1 = Enable PV alarm limit (high) | BOOL | 1 |
| PV_AH_Lim | Limit PV alarm (high) | REAL | 95.0 |
| PV_AH_MsgEn | 1 = Enable PV alarm (high) message | BOOL | 1 |
| PV_AL_En | 1 = Enable PV alarm limit (low) | BOOL | 1 |
| PV_AL_Lim | PV alarm limit (low) | REAL | 5.0 |
| PV_AL_MsgEn | 1 = Enable PV alarm (low) message | BOOL | 1 |
| PV_Hyst | Hysteresis for PV alarm, warning and tolerance limits | REAL | 1.0 |
| PV_OpScale | Limit for scale in PV bar graph of faceplate | STRUCT | - |
| | | • High: REAL | • 100.0 |
| | | • Low: REAL | • 0.0 |
| PV_T_DC* | Delay time for incoming PV tolerance messages [s] | REAL | 0.0 |
| PV_T_DG* | Delay time for outgoing PV tolerance messages [s] | REAL | 0.0 |
| PV_TH_En | 1 = Enable PV tolerance limit (high) | BOOL | 0 |
| PV_TH_Lim | Limit PV tolerance message (high) | REAL | 85.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_TH_MsgEn | 1 = Enable message for PV tolerance message (high) | BOOL | 1 |
| PV_TL_En | 1 = Enable PV tolerance limit (low) | BOOL | 0 |
| PV_TL_Lim | Limit PV tolerance message (low) | REAL | 15.0 |
| PV_TL_MsgEn | 1 = Activate message for PV tolerance message (low) | BOOL | 1 |
| PV_Unit | Unit of measure for process value | INT | 1001 |
| PV_W_DC* | Delay time for incoming PV warnings [s] | REAL | 0.0 |
| PV_W_DG* | Delay time for outgoing PV warnings [s] | REAL | 0.0 |
| PV_WH_En | 1 = Enable PV warning limit (high) | BOOL | 1 |
| PV_WH_Lim | Limit PV warning (high) | REAL | 90.0 |
| PV_WH_MsgEn | 1 = Enable PV warning (high) message | BOOL | 1 |
| PV_WL_En | 1 = Enable PV warning limit (low) | BOOL | 1 |
| PV_WL_Lim | Limit PV warning (low) | REAL | 10.0 |
| PV_WL_MsgEn | 1 = Enable PV warning (low) message | BOOL | 1 |
| RackNo | Rack number | BYTE | 16#FF |
| Rbk* | Position feedback for display on OS | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| RbkHyst | Alarm hysteresis for position feedback | REAL | 1.0 |
| RbkWH_En | 1 = Enable warning (high) for position feedback | BOOL | 1 |
| RbkWH_Lim | Limit for position feedback of warning (high) | REAL | 100.0 |
| RbkWH_MsgEn | 1 = Enable messages for warning (high) for position feedback | BOOL | 1 |
| RbkWL_En | 1 = Enable warning (low) for position feedback | BOOL | 1 |
| RbkWL_Lim | Limit for position feedback of warning (low) | REAL | 0.0 |
| RbkWL_MsgEn | 1 = Enable messages for warning (low) for position feedback | BOOL | 1 |
| RefStdDevIn | Reference value of PV standard deviation (sigma) in defined "good" state of control loop | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#78 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| S_RbkOnPIDTun | Simulation of position feedback on;<br>For PCS 7 PID tuner only | BOOL | 0 |
| S_RbkPIDTun | Simulated position feedback | REAL | 50.0 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | - |
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SimPV* | Process value used for SimOn = 1 | REAL | 0.0 |
| SimPV_Li | Process value that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SimRbk* | Position feedback used for SimOn = 1 | REAL | 0.0 |
| SimRbkLi | Position feedback used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SlotNo | Slot number | BYTE | 16#FF |
| SP_DnRaLim | Limit (low) for the gradient of the set-point [SP_Unit/s] | REAL | 100.0 |
| SP_ExHiLim* | Limit (high) for external setpoint | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| SP_ExLoLim* | Limit (low) for external setpoint | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_Ext* | External setpoint of - (to interconnection) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_ExtLi* | 1 = Select external setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtOp* | 1 = Select external setpoint (via operator) | BOOL | 0 |
| SP_InHiLim* | Limit (high) of internal setpoint | REAL | 100.0 |
| SP_InLoLim* | Limit (low) of internal setpoint | REAL | 0.0 |
| SP_Int* | Internal setpoint for operation | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP_IntLi* | 1 = Select internal setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_IntOp* | 1 = Select internal setpoint (via operator) | BOOL | 0 |
| SP_LiOp | Select setpoint source (internal/external):<br>1 = Via interconnection<br>0 = Via operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_RateOn* | 1 = Activate limitation of setpoint gradients | BOOL | 0 |
| SP_RmpModTime | 1 = Use time (SP_RmpTime) for setpoint ramp,<br>0 = Use gradient | BOOL | 0 |
| SP_RmpOn* | 1 = Activate setpoint ramp to target setpoint SP_RmpTarget | BOOL | 0 |
| SP_RmpTarget* | Target setpoint for setpoint ramp | REAL | 0.0 |
| SP_RmpTime* | Time for setpoint ramp [s] from current SP up to SP_RmpTarget | REAL | 0.0 |
| SP_TrkExt | 1 = Bumpless switchover from external to internal setpoint active | BOOL | 1 |
| SP_TrkPV | 1 = Setpoint follows PV in "manual mode" and with tracking | BOOL | 0 |
| SP_UpRaLim | Gradient limit (high) for the setpoint [SP_Unit/s] | REAL | 100.0 |
| StepCon | Controller type in the FM 355 S:<br>0 = Pulse controller<br>1 = Step controller | BOOL | 0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| StopLi* | 1 = Stop via interconnection or CFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopOp* | 1 = Stop via operator | BOOL | 0 |
| SubN1_ID | ID of the primary DP master system | BYTE | 16#FF |
| SubN2_ID | ID of the redundant DP master system | BYTE | 16#FF |
| TD* | Derivative time [s]<br>TD.ST = 16#FF: Enabled in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| TextRef | Text reference | WORD | 0 |
| TI* | Integral time [s]<br>TI.ST = 16#FF: Enabled in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| TimeFactor | Time unit:<br>0 = Seconds<br>1 = Minutes<br>2 = Hours | INT | 0 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### In/out parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| DataXchg | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| EnCoord | Current coordination number | STRUCT<br>• CO_ACT : INT | -<br>• 0 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AdvCoAct | 1 = "Program mode" enabled.<br>The parameter is available on the WinCC OPC server to be read by external APC programs. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AdvCoRdy | 1 = "Program mode" available | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutAct | 1 = "Automatic mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ChFM_Err | 1 = Channel error on the module | BOOL | 0 |
| Close | Control output:<br>1 = Closed is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| EnCoNum | Coordination number | BYTE | 16#00 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ER | Error signal | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ER_AH_Act | 1 = Alarm limit (high) for control deviation violated.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ER_AL_Act | 1 = Alarm limit (low) for control deviation violated.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ErrorNum | Output of pending error number; for error numbers that can be output by this block, see FmCont error handling (Page 892) | INT | -1 |
| FbkClsOut | 1 = Low limit stop of the position feedback reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkOpnOut | 1 = High limit stop of the position feedback reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FFwdOut | Disturbance variable generated in the FM | STRUCT<br>Value: REAL<br>ST: BYTE | -<br>0.0<br>16#80 |
| FuzCon | Controller type:<br>0 = PID controller<br>1 = Fuzzy controller | BOOL | 0 |
| FuzOptAct | 1 = Optimization of fuzzy controller active | BOOL | 0 |
| FuzSP_PV_Act | Fuzzy controller display:<br>Setpoint < actual value | BOOL | 0 |
| GainEff | Effective proportional gain depends on NegGain, Gain, NormPV, and NormMV | REAL | 1.0 |
| GrpErr | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| LoopClosed | 1 = Control loop closed<br>0 = Control loop open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManAct | 1 = "Manual mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| ManARW_Act | 1 = Tracking mode or anti-reset windup by secondary controller | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManHiOut | Limit (high) for "manual mode", corresponds to input parameter ManHiLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| ManLoOut | Limit (low) for "manual mode", corresponds to input parameter ManLoLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ModErr | 1 = Module error | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message acknowledgement status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |
| MsgAckn2 | Message acknowledgement status 2 (output ACK_STATE of second ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | Alarm error 1 (output ERROR of first ALARM_8P) | BOOL | 0 |
| MsgErr2 | Alarm error 2 (output ERROR of second ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| MsgStat2 | Message status 2 (output STATUS of second ALARM_8P) | WORD | 16#0000 |
| MV | Manipulated variable | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MV_FmTrkAct | 1 = Track manipulated variable in the FM enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_HiAct | 1 = Limit (high) of manipulated variable violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_LoAct | 1 = Limit (low) of manipulated variable violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_SafeAct | 1 = Neutral position manipulated variable of the FM enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_SpliA | Manipulated variable A of split-range function | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_SpliB | Manipulated variable B of split-range function | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_UnitOut | Unit of measure for manipulated variable, for interconnecting to the MV_Unit input parameter of the ConPerMon block | INT | 0 |
| MV_Visible | 1 = MV display visible<br>Evaluated by block icon | BOOL | 0 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Open | Control output:<br>1 = Open is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS_Perm1 | DWORD | 16#FFFFFFFF |
| OS1PermLog | Display of OS1Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| ParFM_Err | 1 = Direct parameter-assignment error of the FM or input Channel configured incorrectly | BOOL | 0 |
| PerAccErr | 1 = I/O access error | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV | Process value of the module | STRUCT<br>• Value: REAL<br>• ST:BYTE | -<br>• 0.0<br>• 16#80 |
| PV_AH_Act | 1 = PV alarm (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_AL_Act | 1 = PV alarm (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Out | Output for process value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_TH_Act | 1 = PV tolerance message (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_TL_Act | 1 = PV tolerance message (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_ToleHi | Limit (high) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_ToleLo | Limit (low) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_UnitOut | Unit of measure for process value, for interconnecting to the PV_Unit input parameter of the ConPerMon block | INT | 0 |
| PV_WH_Act | 1 = PV warning (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_WL_Act | 1 = PV warning (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RbkOut | Output for position feedback | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RbkVisible | 1 = Rbk display visible<br>Evaluated by block icon | BOOL | 0 |
| RbkWH_Act | 1 = Warning (high) for position feedback active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RbkWL_Act | 1 = Warning (low) for position feedback active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToStart | 1 = Active start readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RetValue | Return value of WRREC / RDREC | WORD | 16#0000 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP | Setpoint used by controller | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_DnRaAct | 1 = Negative gradient limiting of setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExHiAct | 1 = Limit (high) for external setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExLoAct | 1 = Limit (low) for external setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP_ExtAct | 1 = External setpoint is active<br>0 = Internal setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtOut | External setpoint, corresponds to input parameter SP_Ext | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_InHiOut | Limit (high) for SP_Int corresponds to input parameter SP_InHiLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| SP_InLoOut | Limit (low) for SP_Int corresponds to input parameter SP_InLoLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_RateTarget | Target setpoint for the gradient limitation | REAL | 0.0 |
| SP_UpRaAct | Positive gradient limiting of setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| SplitRange | 1 = Split-range function has been activated | BOOL | 0 |
| Status1 | Status word 1 (Page 876) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 876) | DWORD | 16#00000000 |
| Status3 | Status word 2 (Page 876) | DWORD | 16#00000000 |
| Stop | Control output:<br>1 =Stopped is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SumMsgAct | 1 = Active hardware interrupt | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| WithRbk | Controller type:<br>0 = Step controller without position feedback<br>1 = Step controller with position feedback | BOOL | 0 |

## See also

FmCont messaging (Page 894)

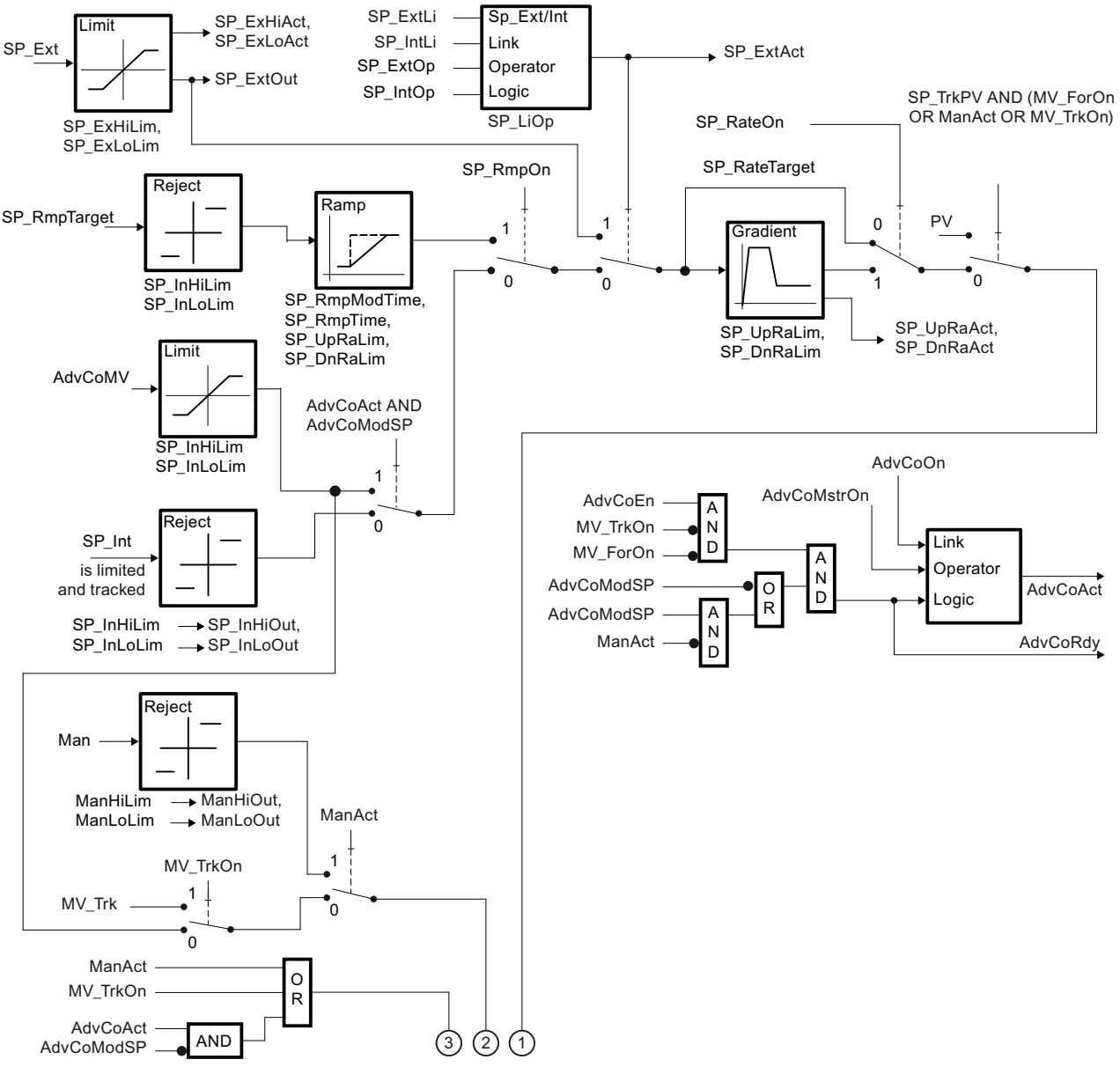
FmCont block diagram (Page 913)

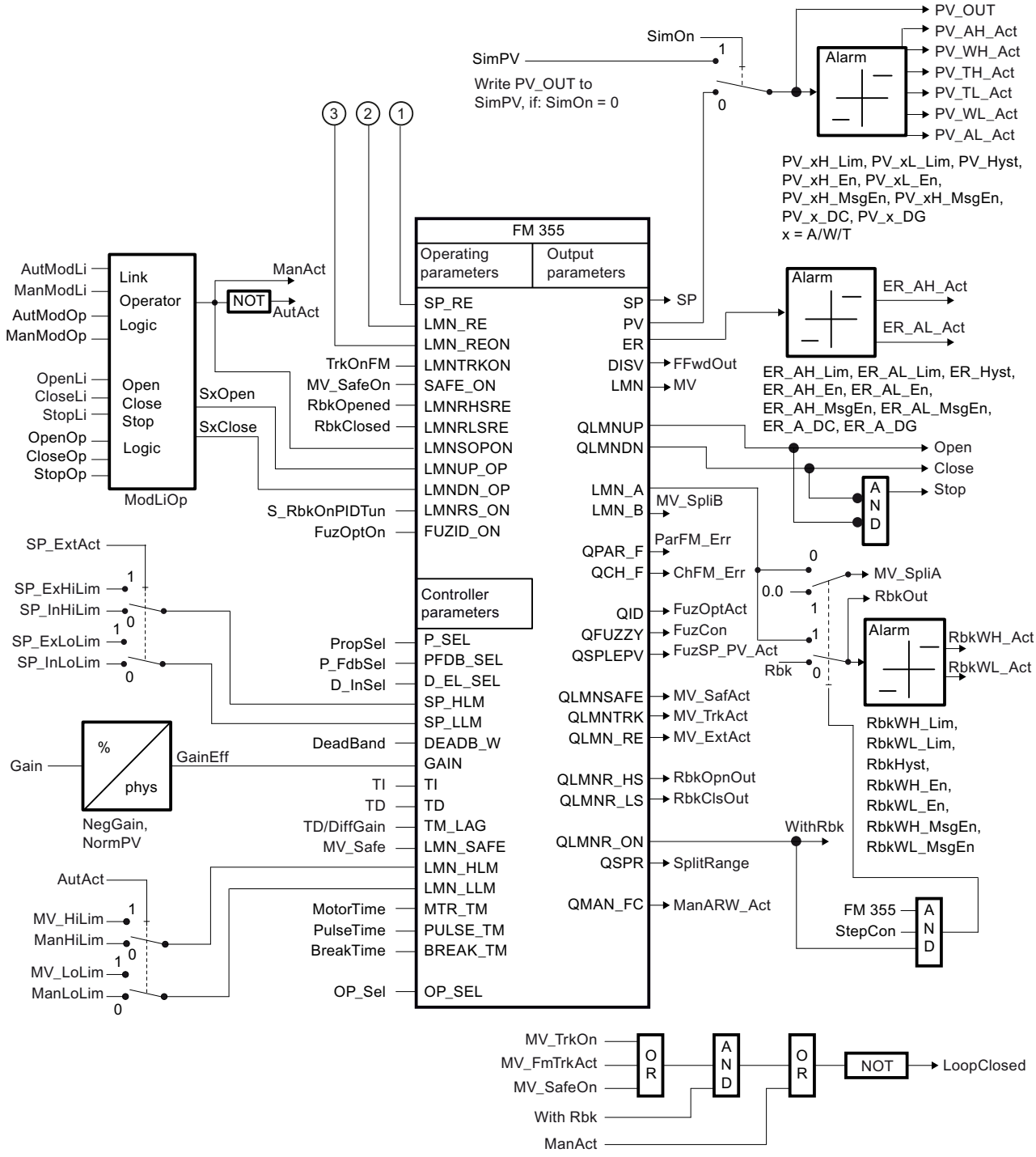
FmCont modes (Page 880)

Neutral position for motors, valves and controllers (Page 54)

## 6.3.7 FmCont block diagram

### FmCont block diagram






## See also

FmCont I/Os (Page 897)

FmCont messaging (Page 894)

FmCont error handling (Page 892)

FmCont functions (Page 882)

FmCont modes (Page 880)

Description of FmCont (Page 876)

## 6.3.8 Operator control and monitoring

### 6.3.8.1 FmCont views

#### Views of the FmCont block

The block FmCont provides the following views:

- FM controllers standard view (analog) (Page 294)
- FM controllers standard view (pulse controller) (Page 298)
- FM controllers standard view (step controller with position feedback) (Page 302)
- FM controllers standard view (step controller without position feedback) (Page 306)
- Alarm view (Page 336)
- Limit view of FM controllers (Page 322)
- Trend view (Page 341)
- Ramp view (Page 334)
- Parameter view of FM controllers (Page 318)
- Preview of FM controllers (Page 331)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icons for PID and FM controller (Page 269)

Refer to the Structure of the faceplate (Page 279) and Block icon structure (Page 260) sections for general information about the faceplate and block icon.

# 6.4 FmTemp - Interface to temperature controller modules FM 355-2

## 6.4.1 Description of FmTemp

### Object name (type + number) and family

Type + number: FB 1819

Family: Control

### Area of application for FmTemp

The block is used for the following applications:

- Fixed setpoint control
- Cascade control
- Ratio control
- Split-range control

### How it works

Block FmTemp is used to interface the FM 355-2 temperature controller modules.

FmTemp can be used for the C (continuous controllers) and S (step and pulse controllers) module types. It contains the algorithms of the setpoint ramp, the setpoint rise limitation, and the limit monitoring of the process value, the control deviation, and the position feedback. Limit monitoring is not used on the module.

The control function itself (e.g. PID algorithm) is processed on the module. You can use the FmTemp block to monitor all relevant process values and to change all relevant controller parameters.

Application examples of the FM 355-2 and detailed descriptions of the associated input and output parameters can be found in the manual of the FM 355-2. temperature controller.

It is primarily used for controlling temperature processes, but can also control level and flow processes which are not excessively fast, for example.

Module FM 355-2 features online optimization of the PID parameters. You can set the corresponding parameters for performing online optimization in the CFC chart at block FmTemp .

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100). Set the input `LogAddr` to the module address from HW Config and the input `Channel` to the desired controller channel (0 ... 3)..

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The parameter CoordNo is set
- The in/out parameter Mode is interconnected to the corresponding OMODE_xx output parameter of the MOD block.
- The parameter FM 355_2 is set in accordance with the module type C/S
- The in/out parameter EnCoord is interconnected to the output EN_CO_x of the FM_CO block of the Basis Library (x = number of the rack)
- The output EnCoNum is interconnected to the input ENCOx_yy of the FM_CO block (x = number of the rack, yy = coordination number).

For the control loop monitoring to work as planned in the trend view of the controller faceplates, the

S7_xarchive:='Value, shortterm;'

attributes in the process tag types for control loops at the controller function block must be set for the following tags:

- Input parameters:
  - CPI_In
- Output parameters
  - MV
  - MV_HiAct
  - MV_LoAct
  - LoopClosed
  - SP
  - PV_Out
  - PV_ToleHi
  - PV_ToleLo

## Startup characteristics

Use the Feature Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at RunUpCyc.

## Status word allocation for Status1 parameter

You can find a description for each parameter in section FmTemp I/Os (Page 938).

| Status bit | Parameter |
|---|---|
| 0 | Occupied |
| 1 | BatchEn |
| 2 | SimOn |

| Status bit | Parameter |
|---|---|
| 3 | OosAct.Value |
| 4 | OosLi.Value |
| 5 | AutAct.Value |
| 6 | Invalid signal status |
| 7 | ManAct.Value |
| 8 | SP_ExtAct.Value |
| 9 | MV_SafeOn.Value |
| 10 | MV_TrkOn.Value |
| 11 | MV.Value > ManLoLim for continuous or pulse controller<br>NOT FbkClosed.Value for step controller with/without position feedback |
| 12 | Open.Value |
| 13 | Close.Value |
| 14 | Stop.Value |
| 15 | FbkOpened.Value |
| 16 | FbkClosed.Value |
| 17 | SimOn AND ManAct |
| 18 | SimOn AND ManAct |
| 19 | AdvCoAct |
| 20 | 1 = Input parameter Rbk is not interconnected (RbkOut.ST = 16#FF) |
| 21 | NegGain |
| 22 - 26 | Not used |
| 27 | AdvCoModSP |
| 28 | 1 = Analog controller (FM 355_2 = 1) |
| 29 | 1 = Pulse controller (FM 355_2 = 0 AND StepCon = 0) |
| 30 | 1 = Step controller with position feedback<br>(FM 355_2 = 0 AND StepCon = 1 AND WithRbk = 1 ) |
| 31 | 1 = Step controller without position feedback<br>(FM 355_2 = 0 AND StepCon = 1 AND WithRbk = 0 ) |

### Status word allocation for Status2 parameter

| Status bit | Parameter |
|---|---|
| 0 | MsgLock |
| 1 | PV_AH_Act.Value |
| 2 | PV_WH_Act.Value |
| 3 | PV_TH_Act.Value |
| 4 | PV_TL_Act.Value |
| 5 | PV_WL_Act.Value |
| 6 | PV_AL_Act.Value |
| 7 | PV_AH_En OR Feature.Bit31 |
| 8 | PV_WH_En OR Feature.Bit31 |
| 9 | PV_TH_En OR Feature.Bit31 |

| Status bit | Parameter |
|---|---|
| 10 | PV_TL_En OR Feature.Bit31 |
| 11 | PV_WL_En OR Feature.Bit31 |
| 12 | PV_AL_En OR Feature.Bit31 |
| 13 | PV_AH_MsgEn |
| 14 | PV_WH_MsgEn |
| 15 | PV_TH_MsgEn |
| 16 | PV_TL_MsgEn |
| 17 | PV_WL_MsgEn |
| 18 | PV_AL_MsgEn |
| 19 | ER_AH_Act.Value |
| 20 | ER_AL_Act.Value |
| 21 | ER_AH_En OR Feature.Bit31 |
| 22 | ER_AL_En OR Feature.Bit31 |
| 23 | ER_AH_MsgEn |
| 24 | ER_AL_MsgEn |
| 25 | RbkWH_Act.Value |
| 26 | RbkWL_Act.Value |
| 27 | RbkWH_En OR Feature.Bit31 |
| 28 | RbkWL_En OR Feature.Bit31 |
| 29 | RbkWH_MsgEn |
| 30 | RbkWL_MsgEn |
| 31 | MS_RelOp |

### Status word allocation for Status3 parameter

| Status bit | Parameter |
|---|---|
| 0 | Effective signal 1 of the message block connected via EventTsIn |
| 1 | Effective signal 2 of the message block connected via EventTsIn |
| 2 | Effective signal 3 of the message block connected via EventTsIn |
| 3 | Effective signal 4 of the message block connected via EventTsIn |
| 4 | Effective signal 5 of the message block connected via EventTsIn |
| 5 | Effective signal 6 of the message block connected via EventTsIn |
| 6 | Effective signal 7 of the message block connected via EventTsIn |
| 7 | Effective signal 8 of the message block connected via EventTsIn |
| 8 | Delay of the PV_AH_Lim message |
| 9 | Delay of the PV_WH_Lim message |
| 10 | Delay of the PV_TH_Lim message |
| 11 | Delay of the PV_TL_Lim message |
| 12 | Delay of the PV_WL_Lim message |
| 13 | Delay of the PV_AL_Lim message |
| 14 | Delay of the ER_AH_Lim message |
| 15 | Delay of the ER_AL_Lim message |

| Status bit | Parameter |
|---|---|
| 16 | Collection of message delays |
| 17 - 19 | Not used |
| 20 | 1= Suppressed messages |
| 21 - 23 | Not used |
| 24 | Internal error |
| 25-26 | Not used |
| 27 | `SP_UpRaAct`, `SP_DnRaAct` limits enabled for gradient mode (`SP_RateOn = 1`) |
| 28 | `GrpErr.Value` |
| 29 | `RdyToStart.Value` |
| 30 | `SimLiOp.Value` |
| 31 | Not used |

### Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 - 7 | Effective signal 9..16 of the message block connected via `EventTsIn` |
| 8 - 31 | Not used |

### See also

FmTemp functions (Page 922)

FmTemp messaging (Page 935)

FmTemp modes (Page 920)

FmTemp error handling (Page 934)

FmTemp block diagram (Page 956)

## 6.4.2 FmTemp modes

### FmTemp operating modes

The block can be operated using the following modes:

- Automatic mode (Page 79)
- Manual mode (Page 79)
- Program mode for controllers (Page 85)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and bumpless switchover in the Manual and automatic mode for control blocks (Page 79) section.

### "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the Manual and automatic mode for control blocks (Page 79) section.

### "Program mode for controllers"

General information on "Program mode for controllers" is available in the section Program mode for controllers (Page 85).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit5` =1 and `Feature.Bit5` =0 is as shown below:

| Output Parameters | **`Feature.Bit5` =0** | **`Feature.Bit5` =1** |
|---|---|---|
| RbkOut.ST | 16#80 | 16#60 |
| RbkWH_Act.ST | 16#XX | 16#60 |
| RbkWL_Act.ST | 16#XX | 16#60 |
| FbkOpnOut.ST | 16#80 | 16#60 |
| FbkClsOut.ST | 16#80 | 16#60 |
| SP_ExtAct.ST | 16#80 | 16#60 |
| SP.ST | 16#80 | 16#60 |
| SP_ExtOut.ST | 16#80 | 16#60 |
| PV.ST | 16#80 | 16#60 |
| PV_Out.ST | 16#80 | 16#60 |
| PV_AH_Act.ST | 16#XX | 16#60 |
| PV_WH_Act.ST | 16#XX | 16#60 |
| PV_TH_Act.ST | 16#XX | 16#60 |
| PV_TL_Act.ST | 16#XX | 16#60 |
| PV_WL_Act.ST | 16#XX | 16#60 |
| PV_AL_Act.ST | 16#XX | 16#60 |
| PV_ToleHi.ST | 16#XX | 16#60 |
| PV_ToleLo.ST | 16#XX | 16#60 |
| ER.ST | 16#80 | 16#60 |
| ER_AH_Act.ST | 16#XX | 16#60 |
| ER_AL_Act.ST | 16#XX | 16#60 |

### See also

Description of FmTemp (Page 916)
FmTemp functions (Page 922)
FmTemp error handling (Page 934)
FmTemp messaging (Page 935)
FmTemp I/Os (Page 938)
FmTemp block diagram (Page 956)

## 6.4.3 FmTemp functions

### Functions of FmTemp

The functions for this block are listed below.

### Module types

FmTemp can be used for the C (continuous controllers) and S (step controllers with and without position feedback and pulse controllers) module types. You can use the following parameters to identify which module type and controller type has been set:

| FM 355 | StepCon | WithRbk | Module type, controller type |
|---|---|---|---|
| 1 or C | - | - | FM 355-2 C: Continuous controller |
| 0 or S | 1 | 1 | FM 355-2 S: Step controller with position feedback |
| 0 or S | 1 | 0 | FM 355-2 S: Step controller without position feedback |
| 0 or S | 0 | - | FM 355-2 S: Pulse controller |

### Generating manipulated variables for continuous controllers, step controllers with position feedback, or pulse controllers

The manipulated variable MV and the actuating signals Open, Close and Stop are generated as follows:

| MV_Safe-On | MV_FMTrk On | ManAct | MV_TrkOn | AdvCoAct AND NOT AdvCo-ModSP | MV = | Manipu-lated variable limit | State | Open, Close, Stop |
|---|---|---|---|---|---|---|---|---|
| 1 | - | - | - | - | MV_Safe | MV_HiLim<br>MV_LoLim | Tracking to safety value | **Cont. controller:** Open, Close, Stop = 0<br>**Step controller with position feedback:** Depending on Rbk and MV, the output signals Open, Close and Stop are generated using the algorithm of a positioner.<br>**Pulse controller:** Depending on MV, the output signals Open and Close are generated using the algorithm of a pulse controller (Stop = 0). |
| 0 | 1 | - | - | - | Prepared FM analog input | MV_HiLim<br>MV_LoLim | Tracking to an FM analog input | |
| 0 | 0 | 1 | - | - | Man | ManHiLim<br>ManLoLim | Manual mode, set by the operator | |
| 0 | 0 | 0 | 1 | - | MV_Trk | MV_HiLim<br>MV_LoLim | Tracking to block input MV_Trk | |
| 0 | 0 | 0 | 0 | 1 | AdvCoMV | MV_HiLim<br>MV_LoLim | Higher-level program mode | |
| 0 | 0 | 0 | 0 | 0 | P_Part + I_Part + D_Part + FFwd | MV_HiLim<br>MV_LoLim | Automatic mode (PID algorithm) | |

### Generating actuating signals for step controllers without position feedback (WithRbk = 0)

The manipulated variable signals `Open`, `Close` and `Stop` can be generated as follows:

| ManAct | Open, Close, Stop | State | ManAct |
|---|---|---|---|
| 1 | The output signals are generated using input signals `OpenOp`/`Li`, CloseOp/`Li` or `StopOp`/`Li` | Manual mode, set by the operator | 1 |
| 0 | The output signals are generated using PID output variables `P_Part`, `I_Part`, `D_Part` and `FFwd` | Automatic mode (PID algorithm) | 0 |

### Tracking and limiting a manipulated variable (cont. controller, step controller with position feedback and pulse controller)

The block provides the standard function Tracking and limiting a manipulated variable (Page 222).

### Neutral position

The controller modules have their own mechanism for feedforwarding a safety value (see manual for Temperature Controller FM 355-2).

### Group error

This block provides the standard function Outputting group errors (Page 130).

The following parameters are taken into consideration when forming the group error:

- `CSF`
- `ModErr`
- `ParFM_Err`
- `PerAccErr`

### Outputting a signal for start readiness

This block provides the standard function Outputting a signal for start readiness (Page 59).

### "Actuator active" information

For continuous and pulse controllers: If the manipulated variable `MV` is greater than the minimum manual limit `ManLoLim`, this is recognized as actuator active.

For step controllers: If the parameter `FbkClosed` = 0 , this is recognized as "Actuator active".

This status can be used to indicate a customized symbol in the process image, for example, and is saved in the status word (see Status word section in Description of FmTemp (Page 916)).

### Limit monitoring of position feedback (cont. controller, step controller with position feedback and pulse controller)

The block provides the standard function Limit monitoring of the feedback (Page 101).

### Group display SumMsgAct for limit monitoring, CSF and ExtMsgx

The block provides the standard function Group display for limit monitoring, CSF and ExtMsgx (Page 92).

### External/internal setpoint specification

The block provides the standard function Setpoint specification - internal/external (Page 136).

### Setpoint limiting for external setpoints

The block provides the standard function Setpoint limiting for external setpoints (Page 222).

### Limitation of rate of change of setpoint

The block provides the standard function Gradient limit of the setpoint (Page 132).

### Using setpoint ramp

The block provides the standard function Using setpoint ramp (Page 131).

### Tracking setpoint in manual mode

The block provides the standard function Tracking setpoint in manual mode (Page 222).

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Process value (`SimPV`, `SimPV_Li`)
- Position feedback (`SimRbk`, `SimRbkLi`)

**Note**

The simulated process value `SimPV` only affects alarm processing and not the PID algorithm in the control module.

### Limit monitoring of the process value

The block provides the standard function Limit monitoring of the process value (Page 93).

### Error signal generation and dead band

The block provides the standard function Error signal generation and dead band (Page 217).

### Limit monitoring of error signal

The block provides the standard function Limit monitoring of setpoint, manipulated variable and control deviation (Page 102).

### Inverting control direction

The block provides the standard function Inverting control direction (Page 217).

### Physical standardization of setpoint, manipulated variable and process value

Controller gain `Gain` is entered either using a physical variable or as standardized value.

- `Gain` as a physical variable:
  The standardized variables retain their default values:
  - `NormPV.High` = 100 and `NormPV.Low` = 0
  - `NormMV.High` = 100 and `NormMV.Low` = 0

  For step controllers with/without position feedback and pulse controllers, the values of `NormMV.High` and `NormMV.Low` are not taken into account. The algorithm uses default values 0 and 100 for internal calculations.
  The effective gain is: `GainEff` = `Gain`
- Entering a standardized `Gain` (dimensionless):
  Change the standardized variables to the actual range of the process values and manipulated variables.
  - Internal and external setpoints; the process value and corresponding parameters are entered according to the physical measuring range of the process value.

  Continuous controller, pulse controller:
  - The manual value, the tracking value of the manipulated variable, disturbance variable feedforward and the corresponding parameters are set according to the physical measuring range of the manipulated variable.

  Step controller with position feedback:
  - The manual parameter, the tracking value of the manipulated variable, disturbance variable feedforward and the corresponding parameters are entered as a percentage 0 ... 100.

  Step controller without position feedback:
  - No physical measuring range available.

  The effective gain is:
  - Step controller with/without position feedback:
    `GainEff` = 100.0 / (`NormPV.High` - `NormPV.Low`) · `Gain`
  - Continuous controller, pulse controller:
    `GainEff` = (`NormMV.High` - `NormMV.Low`) / (`NormPV.High` - `NormPV.Low`) · `Gain`

### Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 240).

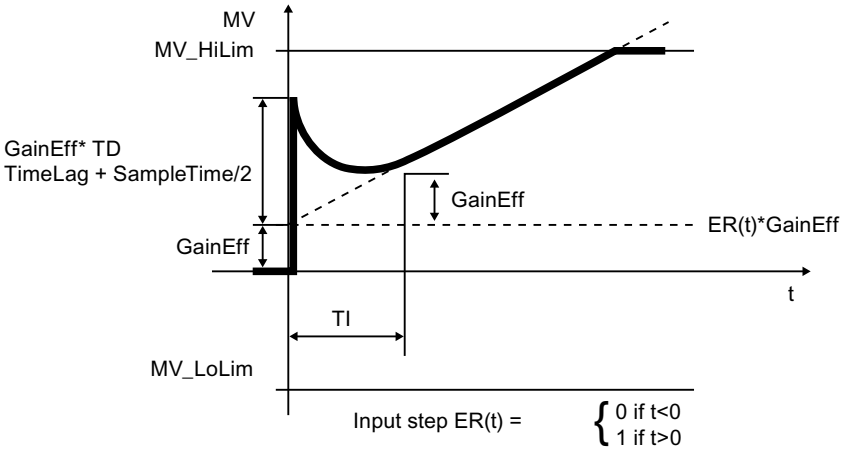
### PID algorithm

The manipulated variable is generated in automatic mode according to the following algorithm:

`MV` = `GainEff` · (1 + 1 / (`TI` · s) + (`TD` · s) / (1 + `TD` / `DiffGain` · s)) · `ER`

Where: s = Complex number

The following step response occurs:




**Note**

This formula describes a standard application where P, I and D components are activated and the P and D components are not in the feedback circuit (`PropSel` = 1, `TI` ≠ 0, `D_InSel` = 0 and `PropFacSP` = 1).

The D component delay is derived from `TD` / `DiffGain`.

- The P component can be shut down by `PropSel` = 0 .
- The I component can be shut down by `TI` = 0 .
- The D component can be shut down by `TD` = 0 .

### Structure segmentation at controllers

The PID controller algorithm of FM 355 features structure segmentation. It is activated via the `PropFacSel` and `D_InSel` parameters. The precise functionality is described in the FM 355 manual.

### Online optimization of the PID controller parameters

- Optimization sequence
  The optimization sequence is as follows:
  - Create a steady state
  - Set PID_On = 1 (if PID parameters are required)
  - Assign parameters of TunD_MV / TunC_MVLMN
  - Set TunOn = 1 (phase 1, preparing for optimization)
  - Start the optimization using a step change in the setpoint or by setting TunStart

  If you have not made any configuration errors, the controller optimization is now in phase 2 and StatusH is 0.
  - When the point of inflection has been reached (PHASE ≥ 3) evaluate the diagnostics display at the StatusH parameter. Phase 0 is reached in a few cycles for process type I and the optimization is completed in full. For process types II and III, the optimization goes to phase 7 (checking the process type). If StatusH > 20000 , an estimation error has occurred or the point of inflection has not been reached. In this case, repeat the procedure.
- Result
  - Once optimization is complete, the parameters PropFacSP, GAIN, TI, TD, DiffGain, ConZone are updated (both for the module and at FmTemp). Furthermore, the PI or PID parameter sets are saved on the FM 355-2.
  - The precise procedure is described in the FM 355-2 manual of the temperature controller module.
- Permanent backup of optimized controller parameters
  - Save, compile and download the hardware configuration; the optimized controller parameters are now in the system data block (SDB).
  - Transfer the modified parameters to the offline data management of the CFC via Chart > Readback.

### Anti-windup

The PID control algorithm of FM 355 has an anti-windup function. The I component is frozen or tracked after the manipulated variable has reached its limits (MV_HiLim or MV_LoLim).

### Feedforwarding and limiting disturbance variables

The block provides a function for activating the disturbance variable feedforward. The precise functionality is described in the FM 355-2 manual.

### Control zone

The block provides the standard function Using control zones (Page 220).

### Forming the signal status for blocks

The block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

- Signal status for the setpoint value SP:
  The signal status of the SP output parameter is always equivalent to the signal status of input parameter SP_Ext or SP_Int, depending on how the setpoint is specified. If the internal setpoint SP_Int is used, the signal status is always output as 16#80.
- Signal status for PV_Out, RbkOut, Open, Close, Stop:
  When the manipulated value (for example, substitute value, simulation, last valid value) is activated, the signal status is always 16#60.
  When the ModErr.Value, ChFM_Err, ParFM_Err module error occurs, the signal status of PV_Out is always 16#0. RbkOut is always 16#0 for step controllers with position feedback
  Otherwise, the following applies:

```
PV_Out.ST: 16#80
Step controller: RbkOut.ST: = 16#80
Continuous controller or pulse controller: RbkOut.ST: = Rbk.ST
Open.ST := 16#80;
Close.ST := 16#80;
Stop.ST := 16#80;
```

- Signal status of the error signal ER:
  The signal status of output parameter ER is obtained from the worst signal status of the two output parameters PV_Out and SP and is output. The signal status 16#60 (external manipulated value - for example, substitute value, simulation, last valid value) is suppressed because the block acts as a sink with external simulation.
  Signal status for FbkOpnOut, FbkClsOut:

```
FbkOpnOut.ST := FbkOpened.ST;
FbkClsOut.ST := FbkClosed.ST;
```

- Signal status for the manipulated variable `MV`:
  The status signal from the output parameter `MV` is always set to `16#80` in "manual mode" and for step controllers without position feedback.
  In "automatic mode", the signal status for continuous controllers or pulse controllers is formed from the following parameters:
  `RbkOut.STFFwdOut.STER.ST`With step controllers, the `FbkOpnOut.STFbkClsOut.ST`parameters are also included. The signal status `16#60` (external manipulated value - for example, substitute value, simulation, last valid value) is suppressed because the block acts as a sink with an external manipulated value (for example, substitute value, simulation, last valid value).
- Worst signal status:
  The worst signal status `ST_Worst` for the block is formed from:
  - `PV_Out.ST;`
  - `SP:ST;`
  - `FFwdOut.ST;`
  - `RbkOut.ST;`

  With step controllers (FM355-2 = 0, StepCon = 1), the following are also included:
  - `FbkOpnOut.ST;`
  - `FbkClsOut.ST;`

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 4 | Setting switch or button mode (Page 177) |
| 5 | Changes signal status of outputs in OOS (Page 208) |
| 6 | Ramp rate calculation (Page 191) |
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 23 | SP following PV in open loop has no priority over SP_Ext and SP limits (Page 193) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 28 | Disabling operating points (Page 156) |
| 29 | Signaling limit violation (Page 181) |
| 31 | Limits disabled in faceplate (Page 206) |

### Configurable reactions using the Feature2 parameter

| Bit | Function |
|---|---|
| 11 | Suppress MsgLock and “Out of service” mode for a connected message block (Page 202) |
| 12 | Disable limit act output when limit value message is disabled (Page 209) |

### Operator permissions

The block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the OS_Perm parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" AutModOp |
| 1 | 1 = Operator can switch to "manual mode" ManModOp |
| 2 | 1 = Operator can switch to "Out of service" mode OosOp |
| 3 | 1 = Operator can switch to "program mode" AdvCoEn |
| 4 | 1 = Operator can switch the setpoint to "external" SP_ExtOp |
| 5 | 1 = Operator can switch the setpoint to "internal" SP_IntOp |
| 6 | 1 = Operator can change the internal setpoint SP_Int |
| 7 | Continuous controllers, pulse controllers or step controllers with position feedback:<br>1 = Operator can change the manual parameter Man<br>Step controller without position feedback:<br>1 = Operator can change the manual operation signals OpenOp, StopOp, CloseOp |
| 8 | 1 = Operator can change operation high limit of the setpoint SP_InHiLim |
| 9 | 1 = Operator can change operation low limit of the setpoint SP_InLoLim |
| 10 | 1 = Operator can change the operation high limit of the manipulated variable ManHiLim |
| 11 | 1 = Operator can change the operation low limit of the manipulated variable ManLoLim |
| 12 | 1 = Operator can enable the setpoint's gradient limitation function SP_RateOn |
| 13 | 1 = Operator can change the setpoint's high limit for the ramp SP_UpRaLim |
| 14 | 1 = Operator can change the setpoint's low limit for the ramp SP_DnRaLim |
| 15 | 1 = Operator can switch between the time value or the value for the ramp SP_RmpModTime |
| 16 | 1 = Operator can change the ramp time SP_RmpTime |
| 17 | 1 = Operator can change the target setpoint SP_RmpTarget for the setpoint ramp |
| 18 | 1 = Operator can enable the setpoint ramp function SP_RmpOn |
| 19 | Not used |
| 20 | 1 = Operator can enable the track setpoint in manual mode function SP_TrkPV |
| 21 | 1 = Operator can enable the bumpless switchover from external to internal SP_TrkExt |
| 22 | 1 = Operator can change the gain parameter Gain |
| 23 | 1 = Operator can change the integral time parameter TI |
| 24 | 1 = Operator can change the derivative time parameter TD |
| 25 | 1 = Operator can change the derivative gain parameter DiffGain |
| 26 | 1 = Operator can change the dead band parameter DeadBand |
| 27 | 1 = Operator can change the control zone parameter ConZone |
| 28 | 1 = Operator can change the integral time parameter MotorTime |

| Bit | Function |
|---|---|
| 29 | 1 = Operator can change the integral time parameter `PulseTime` |
| 30 | 1 = Operator can change the integral time parameter `BreakTime` |
| 31 | Not used |

The block has the following permissions for the `OS1Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can change the limit (process value) `PV_AH_Lim` for the high alarm |
| 1 | 1 = Operator can change the limit (process value) `PV_WH_Lim` for the high warning |
| 2 | 1 = Operator can change the limit (process value) `PV_TH_Lim` for the high tolerance |
| 3 | 1 = Operator can change the hysteresis (process value) `PV_Hyst` |
| 4 | 1 = Operator can change the limit (process value) `PV_TL_Lim` for the low tolerance |
| 5 | 1 = Operator can change the limit (process value) `PV_WL_Lim` for the low warning |
| 6 | 1 = Operator can change the limit (process value) `PV_AL_Lim` for the low alarm |
| 7 | 1 = Operator can change the limit (error signal) `ER_AH_Lim` for the high alarm |
| 8 | 1 = Operator can change the hysteresis (error signal) `ER_Hyst` |
| 9 | 1 = Operator can change the limit (error signal) `ER_AL_Lim` for the low alarm |
| 10 | 1 = Operator can change the limit (position feedback) `RbkWH_Lim` for the high warning |
| 11 | 1 = Operator can change the hysteresis (position feedback) `RbkHyst` |
| 12 | 1 = Operator can change the limit (position feedback) `RbkWL_Lim` for the low warning |
| 13 | 1 = Operator can open the valve |
| 14 | 1 = Operator can close the valve |
| 15 | 1 = Operator can stop the valve |
| 16 | 1 = Operator can activate the Simulation function `SimOn` |
| 17 | 1 = Operator can activate the Release for maintenance function `MS_RelOp` |
| 18 | 1 = Operator can activate / deactivate messages via `PV_AH_MsgEn` |
| 19 | 1 = Operator can activate / deactivate messages via `PV_WH_MsgEn` |
| 20 | 1 = Operator can activate / deactivate messages via `PV_TH_MsgEn` |
| 21 | 1 = Operator can activate / deactivate messages via `PV_TL_MsgEn` |
| 22 | 1 = Operator can activate / deactivate messages via `PV_WL_MsgEn` |
| 23 | 1 = Operator can activate / deactivate messages via `PV_AL_MsgEn` |
| 24 | 1 = Operator can activate / deactivate messages via `ER_AH_MsgEn` |
| 25 | 1 = Operator can activate / deactivate messages via `ER_AL_MsgEn` |
| 26 | 1 = Operator can activate / deactivate messages via `RbkWH_MsgEn` |
| 27 | 1 = Operator can activate / deactivate messages via `RbkWL_MsgEn` |
| 28 | 1 = Operator can change the simulation value `SimPV` |
| 29 | 1 = Operator can change the simulation value `SimRbk` |
| 30 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Release for maintenance

The block provides the standard function Release for maintenance (Page 70).

### Generating instance-specific messages

The block provides the standard function Generating instance-specific messages (Page 230) without the time stamp function in the I/O.

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Specifying the display area for process and setpoint values as well as operations

This block provides the standard function Display and operator input area for process values and setpoints (Page 233).

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Button labels

This block provides the standard function Labeling of buttons and text (Page 235)

Instance-specific text can be configured for the following parameters:

- `OpenOp`
- `StopOp`
- `CloseOp`

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### See also

FmTemp messaging (Page 935)

FmTemp I/Os (Page 938)

FmTemp modes (Page 920)

FmTemp block diagram (Page 956)

FmTemp error handling (Page 934)

EventTs functions (Page 2046)

## 6.4.4 FmTemp error handling

### Error handling of FmTemp

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Control system fault (CSF)

### Overview of error numbers

The `ErrorNum` output parameter can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. |
| 30 | The value of `PV` can no longer be displayed in the REAL number field. |
| 31 | The value of `SP_Ext` can no longer be displayed in the REAL number field. |
| 33 | The value of `MV_Trk` can no longer be displayed in the REAL number field. |
| 35 | The value of `Rbk` can no longer be displayed in the REAL number field. |
| 36 | The value of `MV` can no longer be displayed in the REAL number field. |
| 50 | The controller cannot be switched to program mode, because program mode with default setpoint (`AdvCoModSP` = 0) is not possible for step controllers without position feedback (`WithRbk` = 0). |
| 51 | `AutModLi` = 1 and `ManModLi` = 1<br>`SP_LiOp` = 1 and `SP_IntLi` = 1 and `SP_ExtLi` = 1<br>`OpenLi` = 1 and `StopLi` = 1<br>`CloseLi` = 1 and `StopLi` = 1<br>`OpenLi` = 1 and `CloseLi` = 1 |
| 59 | = 1, "Gain is negative" |
| 60 | \|`TI`\| < `SampleTime` / 2 |
| 61 | \|`TD`\| < `SampleTime` |
| 62 | `DiffGain` < 1 or `DiffGain` > 10 |
| 63 | `TD` / `DiffGain` < `SampleTime` / 2 |
| 64 | `PropFacSP` < 0 or `PropFacSP` > 1 |
| 66 | `NormPV_High` = `NormPV_Low` |
| 67 | `MotorTime` < `SampleTime` |

| Error number | Meaning of the error number |
|---|---|
| 68 | PulseTime < SampleTime |
| 69 | BreakTime < SampleTime |
| 70 | Channel < 0 or Channel > 3 |
| 71 | (D_InSel < 0 or D_InSel > 4) and D_InSel ≠ 17 |
| 74 | ConZone < 0.0<br>FM 355-2 sets the output ParFM_Err = 1 |

### Control system fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 128) section for more on this.

### See also

Description of FmTemp (Page 916)

FmTemp modes (Page 920)

FmTemp functions (Page 922)

FmTemp messaging (Page 935)

FmTemp I/Os (Page 938)

FmTemp block diagram (Page 956)

Setting switch or button mode (Page 177)

## 6.4.5 FmTemp messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Process messages
- Instance-specific messages

### Process control fault

The following control system error messages can be output:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId2 | SIG 6 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter `CSF`. If it changes to `CSF` = 1, a process control fault is triggered (`MsgEvId2`, SIG 6).

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId1` | SIG 1 | Alarm - high | $$BlockComment$$<br>`PV` - high alarm limit violated |
| | SIG 2 | Warning - high | $$BlockComment$$<br>`PV` - high warning limit violated |
| | SIG 3 | Tolerance - high | $$BlockComment$$<br>`PV` - high tolerance limit violated |
| | SIG 4 | Tolerance - low | $$BlockComment$$<br>`PV` - low tolerance limit violated |
| | SIG 5 | Warning - low | $$BlockComment$$<br>`PV` - low warning limit violated |
| | SIG 6 | Alarm - low | $$BlockComment$$<br>`PV` - low alarm limit violated |
| | SIG 7 | Alarm - high | $$BlockComment$$<br>`ER` - high alarm limit violated |
| | SIG 8 | Alarm - low | $$BlockComment$$<br>`ER` - low alarm limit violated |
| `MsgEvId2` | SIG 7 | Warning - high | $$BlockComment$$<br>`Rbk` - high warning limit violated |
| | SIG 8 | Warning - low | $$BlockComment$$<br>`Rbk` - low warning limit violated |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Instance-specific messages

You can use up to four instance-specific messages with this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId2 | SIG 1 | AS process control message - fault | $$BlockComment$$<br>External message 1<br>Status 16#@5%x@ |
| | SIG 2 | AS process control message - fault | $$BlockComment$$<br>External message 2<br>Status 16#@6%x@ |
| | SIG 3 | AS process control message - fault | $$BlockComment$$<br>External message 3<br>Status 16#@7%x@ |
| | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External message 4<br>Status 16#@8%x@ |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance **MsgEvId1**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |
| 4 | Process value PV_Out |
| 5 | Control deviation ER |
| 6 | ExtVa106 |
| 7 | ExtVa107 |
| 8 | Not allocated |
| 9 | Not allocated |
| 10 | Not allocated |

The associated values 6 ... 7 are allocated to the parameters ExtVa106 ... ExtVa107 and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### Associated values for message instance **MsgEvId2**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |

| Associated value | Block parameters |
|---|---|
| 4 | Position feedback `RbkOut` |
| 5 | Signal status `ExtMsg1` |
| 6 | Signal status `ExtMsg2` |
| 7 | Signal status `ExtMsg3` |
| 8 | Signal status `ExtMsg4` |
| 9 | `ExtVa209` |
| 10 | `ExtVa210` |

The associated values 9 ... 10 are allocated to the parameters `ExtVa209` ... `ExtVa210` and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of FmTemp (Page 916)

FmTemp functions (Page 922)

FmTemp I/Os (Page 938)

FmTemp modes (Page 920)

FmTemp error handling (Page 934)

FmTemp block diagram (Page 956)

## 6.4.6 FmTemp I/Os

I/Os of FmTemp

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `AccMode*` | 1 = Transfer of operating parameters `SubN1_ID`, `SubN2_ID`, `RackNo`, `SlotNo` and `Channel` to internal processing | BOOL | 1 |
| `AdvCoEn` | 1 = Enable "program mode" via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `AdvCoModSP` | Type of "program mode":<br>1 = Setpoint specification<br>0 = Manipulated variable specification | BOOL | 1 |
| `AdvCoMstrOn` | Activate (0-1) or deactivate (1-0) "program mode" via edge transition | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| AdvCoMV | Specified value from the external program.<br>The parameter is available on the WinCC OPC server to be read by external APC programs. | REAL | 0.0 |
| AdvCoOn* | 1 = Enable "program mode" via faceplate.<br>The parameter is available on the WinCC OPC server to be read by external APC programs. | BOOL | 0 |
| AutModLi* | 1 = "Automatic mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutModOp* | 1 = "Automatic mode" via operator (controlled by ModLiOp = 0) | BOOL | 0 |
| BatchEn | 1 = Enable allocation for batch control | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BreakTime* | Minimum break duration [s] | REAL | 1.0 |
| Channel | Controller channel number (0..3) | INT | 0 |
| CloseLi* | 1 = Close via interconnection or CFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CloseOp* | 1 = Close via operator | BOOL | 0 |
| ConZone* | Control zone | REAL | 0.0 |
| CoordNo | Coordination number | INT | 0 |
| CPI_In | Input for control performance index, which is calculated by the assigned ConPerMon block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#78 |
| CSF | 1 = External error (control system fault) Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| D_InSel* | Input for differentiator:<br>0 = Error signal<br>1..4 = Channel 0..3<br>17 = Actual value to feedback | INT | 0 |
| DeadBand* | Width of dead band | REAL | 0.0 |
| DiffGain* | Gain of differentiator [1..10] DiffGain = TD / (delay time of D component) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 5.0<br>• 16#80 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| ER_A_DC* | Delay for incoming alarms during error signal monitoring | REAL | 0.0 |
| ER_A_DG* | Delay for outgoing alarms during error signal monitoring | REAL | 0.0 |
| ER_AH_En | 1 = Activate alarm (high) for error signal monitoring | BOOL | 1 |
| ER_AH_Lim | Alarm limit (high) for error signal monitoring | REAL | 100.0 |
| ER_AH_MsgEn | 1 = Activate messages for alarm (high) for error signal monitoring | BOOL | 1 |
| ER_AL_En | 1 = Activate alarm (low) for error signal monitoring | BOOL | 1 |
| ER_AL_Lim | Alarm limit (low) for error signal monitoring | REAL | -100.0 |
| ER_AL_MsgEn | 1 = Activate messages for alarm (low) for error signal monitoring | BOOL | 1 |
| ER_Hyst | Alarm hysteresis for error signal | REAL | 1.0 |
| EventTsIn | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ExtMsg3 | Binary input for freely selectable message 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg4 | Binary input for freely selectable message 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| ExtVa209 | Associated value 9 for messages (MsgEvID2) | ANY | |
| ExtVa210 | Associated value 10 for messages (MsgEvID2) | ANY | |
| FbkClosed | Low limit stop signal of position feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkOpened | High limit stop signal of position feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Feature | I/O for additional functions (Page 922) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| FM355_2 | Module type:<br>0: FM 355-2 S;<br>1: FM 355-2 C | BOOL | 0 |
| Gain* | Proportional gain<br>Gain.ST = 16#FF: Enabled in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#FF |
| LoadPID* | Load optimized PI/PID parameters | BOOL | 0 |
| LogAddr | Logical address FM 355 | INT | 0 |
| Man* | Manual specification for the manipulated variable | REAL | 0.0 |
| ManHiLim* | Limit (high) for manual parameter Man | REAL | 100.0 |
| ManLoLim* | Limit (low) for manual parameter Man | REAL | 0.0 |
| ManModLi* | 1 = "Manual mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManModOp* | 1 = "Manual mode" via OS operator (controlled by ModLiOp = 0) | BOOL | 1 |
| Mode | Operating mode | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ModLiOp | Operating mode switchover between:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MotorTime* | Motor actuating time [s] | REAL | 30.0 |
| MS_RelOp* | 1 = Release for maintenance by OS operator | BOOL | 0 |
| MS | Maintenance status | DWORD | 0 |
| MS_Release | Release for maintenance (interconnected with MS_Release of the technology block) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ms_Ext | External maintenance status. For more information, refer to the MS_Ext (Page 2463) parameter section under information on using channel blocks. | DWORD | 0 |
| MsgEvID1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgEvID2 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_FmTrkOn | 1 = Manipulated variable tracking in the FM | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_HiLim* | Limit (high) for manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| MV_LoLim* | Limit (low) for manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_OpScale | OS display range for manipulated variable MV | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| MV_Safe* | Neutral position manipulated variable | REAL | 0.0 |
| MV_SafeOn | 1 = Neutral position manipulated variable MV_Safe at output MV | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_Trk* | Tracking value for the manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MV_TrkOn | 1 = Tracking of manipulated variable MV | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_Unit | Unit of measure for manipulated variable | INT | 1342 |
| NegGain* | 0 = Effective proportional gain GainEff is positive<br>1 = Effective proportional gain GainEff is negative | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NormMV | Manipulated variable range (MV) for standardizing the proportional gain (GAIN) | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| NormPV | Process value range (PV) for standardizing the proportional gain (GAIN) | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| Occupied | Occupied by batch control | BOOL | 0 |
| OosLi | Edge transition (0-1) = "Out of service", via interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpenLi* | 1 = Open via interconnection or CFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpenOp* | 1 = Open via operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 922) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| OS1Perm | I/O for operator permissions (Page 922) | STRUCT<br>• Bit 0: BOOL<br>• Bit 18: BOOL<br>• Bit 19: BOOL<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1<br>• 1 |
| PID_On* | 1 = PID mode on | BOOL | 0 |
| PropFacSP* | Applying the P component to the feedback [0..1].<br>0 = P component fully in feedback | REAL | 1.0 |
| PropSel* | 1 = Activate P component | BOOL | 1 |
| PulseTime* | Minimum pulse duration [s] | REAL | 1.0 |
| PV_A_DC* | Delay time for incoming PV alarms [s] | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_A_DG* | Delay time for outgoing PV alarms [s] | REAL | 0.0 |
| PV_AH_En | 1 = Enable PV alarm limit (high) | BOOL | 1 |
| PV_AH_Lim | Limit PV alarm (high) | REAL | 95.0 |
| PV_AH_MsgEn | 1 = Enable PV alarm (high) message | BOOL | 1 |
| PV_AL_En | 1 = Enable PV alarm limit (low) | BOOL | 1 |
| PV_AL_Lim | PV alarm limit (low) | REAL | 5.0 |
| PV_AL_MsgEn | 1 = Enable PV alarm (low) message | BOOL | 1 |
| PV_Hyst | Hysteresis for PV alarm, warning and tolerance limits | REAL | 1.0 |
| PV_OpScale | Limit for scale in PV bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| PV_T_DC* | Delay time for incoming PV tolerance messages [s] | REAL | 0.0 |
| PV_T_DG* | Delay time for outgoing PV tolerance messages [s] | REAL | 0.0 |
| PV_TH_En | 1 = Enable PV tolerance limit (high) | BOOL | 0 |
| PV_TH_Lim | Limit PV tolerance message (high) | REAL | 85.0 |
| PV_TH_MsgEn | 1 = Enable message for PV tolerance message (high) | BOOL | 1 |
| PV_TL_En | 1 = Enable PV tolerance limit (low) | BOOL | 0 |
| PV_TL_Lim | Limit PV tolerance message (low) | REAL | 15.0 |
| PV_TL_MsgEn | 1 = Activate message for PV tolerance message (low) | BOOL | 1 |
| PV_Unit | Unit of measure for process value | INT | 1001 |
| PV_W_DC* | Delay time for incoming PV warnings [s] | REAL | 0.0 |
| PV_W_DG* | Delay time for outgoing PV warnings [s] | REAL | 0.0 |
| PV_WH_En | 1 = Enable PV warning limit (high) | BOOL | 1 |
| PV_WH_Lim | Limit PV warning (high) | REAL | 90.0 |
| PV_WH_MsgEn | 1 = Enable PV warning (high) message | BOOL | 1 |
| PV_WL_En | 1 = Enable PV warning limit (low) | BOOL | 1 |
| PV_WL_Lim | Limit PV warning (low) | REAL | 10.0 |
| PV_WL_MsgEn | 1 = Enable PV warning (low) message | BOOL | 1 |
| RackNo | Rack number | BYTE | 16#FF |
| RatioFac* | Ratio factor | REAL | 0.0 |
| Rbk* | Position feedback for display on OS | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| RbkHyst | Alarm hysteresis for position feedback | REAL | 1.0 |
| RbkWH_En | 1 = Enable warning (high) for position feedback | BOOL | 1 |
| RbkWH_Lim | Limit for position feedback of warning (high) | REAL | 100.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RbkWH_MsgEn | 1 = Enable messages for warning (high) for position feedback | BOOL | 1 |
| RbkWL_En | 1 = Enable warning (low) for position feedback | BOOL | 1 |
| RbkWL_Lim | Limit for position feedback of warning (low) | REAL | 0.0 |
| RbkWL_MsgEn | 1 = Enable messages for warning (low) for position feedback | BOOL | 1 |
| RefStdDevIn | Reference value of PV standard deviation (sigma) in defined "good" state of control loop | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#78 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SavePar* | 1 = Save PID controller parameters | BOOL | 0 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | - |
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SimPV* | Process value used for SimOn = 1 | REAL | 0.0 |
| SimPV_Li | Process value that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SimRbk* | Position feedback used for SimOn = 1 | REAL | 0.0 |
| SimRbkLi | Position feedback used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SlotNo | Slot number | BYTE | 16#FF |
| SP_DnRaLim | Limit (low) for the gradient of the setpoint [SP_Unit/s] | REAL | 100.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP_ExHiLim* | Limit (high) for external setpoint | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| SP_ExLoLim* | Limit (low) for external setpoint | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_Ext* | External setpoint of - (to interconnection) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_ExtLi* | 1 = Select external setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtOp* | 1 = Select external setpoint (via operator) | BOOL | 0 |
| SP_InHiLim* | Limit (high) of internal setpoint | REAL | 100.0 |
| SP_InLoLim* | Limit (low) of internal setpoint | REAL | 0.0 |
| SP_Int* | Internal setpoint for operation | REAL | 0.0 |
| SP_IntLi* | 1 = Select internal setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_IntOp* | 1 = Select internal setpoint (via operator) | BOOL | 0 |
| SP_LiOp | Select setpoint source (internal/external):<br>1 = Via interconnection<br>0 = Via operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_RateOn* | 1 = Activate limitation of setpoint gradients | BOOL | 0 |
| SP_RmpModTime | 1 = Use time (SP_RmpTime) for setpoint ramp,<br>0 = Use gradient | BOOL | 0 |
| SP_RmpOn* | 1 = Activate setpoint ramp to target setpoint SP_RmpTarget | BOOL | 0 |
| SP_RmpTarget* | Target setpoint for setpoint ramp | REAL | 0.0 |
| SP_RmpTime* | Time for setpoint ramp [s] from current SP up to SP_RmpTarget | REAL | 0.0 |
| SP_TrkExt | 1 = Bumpless switchover from external to internal setpoint active | BOOL | 1 |
| SP_TrkPV | 1 = Setpoint follows PV in "manual mode" and with tracking | BOOL | 0 |
| SP_UpRaLim | Gradient limit (high) for the setpoint [SP_Unit/s] | REAL | 100.0 |
| StepNo | Batch step number | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| StopLi* | 1 = Stop via interconnection or CFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopOp* | 1 = Stop via operator | BOOL | 0 |
| SubN1_ID | ID of the primary DP master system | BYTE | 16#FF |
| SubN2_ID | ID of the redundant DP master system | BYTE | 16#FF |
| TD* | Derivative time [s]<br>TD.ST = 16#FF: Enabled in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| TextRef | Text reference | WORD | 0 |
| TI* | Integral time [s]<br>TI.ST = 16#FF: Enabled in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#FF |
| TimeFactor | Time unit:<br>0 = Seconds<br>1 = Minutes<br>2 = Hours | INT | 0 |
| TunC_MV* | Delta manipulated variable for cooling optimization | REAL | -20.0 |
| TunC_Start* | Start cooling optimization | BOOL | 0 |
| TunD_MV* | Delta manipulated variable for process trigger | REAL | 20.0 |
| TunOn* | Enable controller optimization | BOOL | 0 |
| TunStart* | Start controller optimization | BOOL | 0 |
| UndoPar* | Undo controller parameter changes | BOOL | 0 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## In/out parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| DataXchg | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| EnCoord | Current coordination number | STRUCT<br>• CO_ACT : INT | -<br>• 0 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AdvCoAct | 1 = "Program mode" enabled.<br>The parameter is available on the WinCC OPC server to be read by external APC programs. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AdvCoRdy | 1 = "Program mode" available | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutAct | 1 = "Automatic mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ChFM_Err | 1 = Channel error on the module | BOOL | 0 |
| Close | Control output:<br>1 = Closed is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| EnCoNum | Coordination number | BYTE | 16#00 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ER | Error signal | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ER_AH_Act | 1 = Alarm limit (high) for control deviation violated.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ER_AL_Act | 1 = Alarm limit (low) for control deviation violated.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ErrorNum | Output of pending error number, for error numbers that can be output by this block, see FmTemp error handling (Page 934) | INT | -1 |
| FFwdOut | Disturbance variable generated in the FM | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| FbkClsOut | 1 = Low limit stop of the position feedback reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkOpnOut | 1 = High limit stop of the position feedback reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| GainEff | Effective proportional gain depends on NegGain, Gain, NormPV, and NormMV | REAL | 1.0 |
| GrpErr | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LoopClosed | 1 = Control loop closed<br>0 = Control loop open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManAct | 1 = "Manual mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| ManARW_Act | 1 = Tracking mode or anti-reset windup by secondary controller | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManHiOut | Limit (high) for "manual mode", corresponds to input parameter ManHiLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| ManLoOut | Limit (low) for "manual mode", corresponds to input parameter ManLoLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ModErr | 1 = Module error | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 0 |
| MsgAckn1 | Message acknowledgement status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |
| MsgAckn2 | Message acknowledgement status 2 (output ACK_STATE of second ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | Alarm error 1 (output ERROR of first ALARM_8P) | BOOL | 0 |
| MsgErr2 | Alarm error 2 (output ERROR of second ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| MsgStat2 | Message status 2 (output STATUS of second ALARM_8P) | WORD | 16#0000 |
| MV | Manipulated variable | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_FmTrkAct | 1 = Track manipulated variable in the FM enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_HiAct | 1 = Limit (high) of manipulated variable violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_LoAct | 1 = Limit (low) of manipulated variable violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_SafeAct | 1 = Neutral position manipulated variable of the FM enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_SpliA | Manipulated variable A of split-range function | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_SpliB | Manipulated variable B of split-range function | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MV_UnitOut | Unit of measure for manipulated variable, for interconnecting to the MV_Unit input parameter of the ConPerMon block | INT | 0 |
| MV_Visible | 1 = MV display visible<br>Is evaluated by the block icon | BOOL | 0 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Open | Control output<br>1 = Open is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermLog | Display of OS1Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS_Perm1 | DWORD | 16#FFFFFFFF |
| ParFM_Err | 1 = Direct parameter-assignment error of the FM or input Channel configured incorrectly | BOOL | 0 |
| PerAccErr | 1 = I/O access error | BOOL | 0 |
| Phase | Phase of auto-tuning [0..7] | INT | 0 |
| PV | Process value of the module | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>0.0<br>16#80 |
| PV_AH_Act | 1 = PV alarm (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>0<br>16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_AL_Act | 1 = PV alarm (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Out | Output for process value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_TH_Act | 1 = PV tolerance message (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_TL_Act | 1 = PV tolerance message (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_ToleHi | Limit (high) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_ToleLo | Limit (low) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_UnitOut | Unit of measure for process value, for interconnecting to the PV_Unit input parameter of the ConPerMon block | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_WH_Act | 1 = PV warning (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29(Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_WL_Act | 1 = PV warning (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RbkOut | Output for position feedback | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RbkWH_Act | 1 = Warning (high) for position feedback active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RbkWL_Act | 1 = Warning (low) for position feedback active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RetValue | Return value of WRREC / RDREC | WORD | 16#0000 |
| RbkVisible | 1 = Rbk display visible<br>Is evaluated by the block icon | BOOL | 0 |
| RdyToStart | 1 = Active start readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP | Setpoint used by controller | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_DnRaAct | 1 = Negative gradient limiting of setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExHiAct | 1 = Limit (high) for external setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExLoAct | 1 = Limit (low) for external setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtAct | 1 = External setpoint is active<br>0 = Internal setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtOut | External setpoint, corresponds to input parameter SP_Ext | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_InHiOut | Limit (high) for SP_Int corresponds to input parameter SP_InHiLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| SP_InLoOut | Limit (low) for SP_Int corresponds to input parameter SP_InLoLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_RateTarget | Target setpoint for the gradient limitation | REAL | 0.0 |
| SP_UpRaAct | Positive gradient limiting of setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SplitRange | 1 = Split-range function has been activated | BOOL | 0 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 916) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 916) | DWORD | 16#00000000 |
| Status3 | Status word 2 (Page 916) | DWORD | 16#00000000 |
| StatusC | Status of cooling optimization | INT | 0 |
| StatusD | Status of controller design | INT | 0 |
| StatusH | Status of heating optimization | INT | 0 |

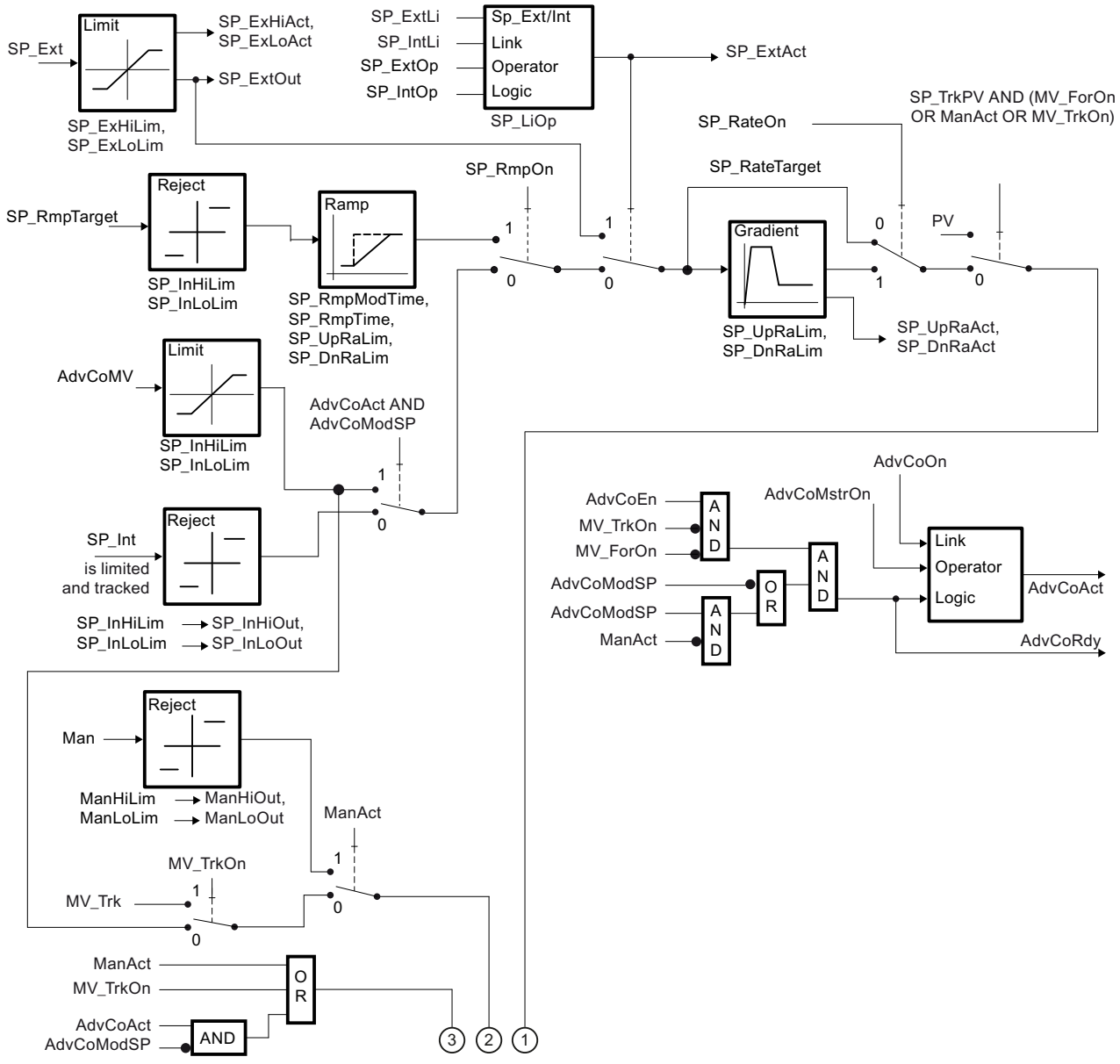
| Parameter | Description | Type | Default |
|---|---|---|---|
| StepCon | 1 = Step controller | BOOL | 0 |
| Stop | Control output:<br>1 =Stopped is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SumMsgAct | 1 = Active hardware interrupt | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| TunAct | 1 = Optimization running | BOOL | 0 |
| WithRbk | Controller type:<br>0 = Step controller without position feedback<br>1 = Step controller with position feedback | BOOL | 0 |
| ZoneTun | Controller channels grouped in one zone for parallel optimization | WORD | 16#0000 |

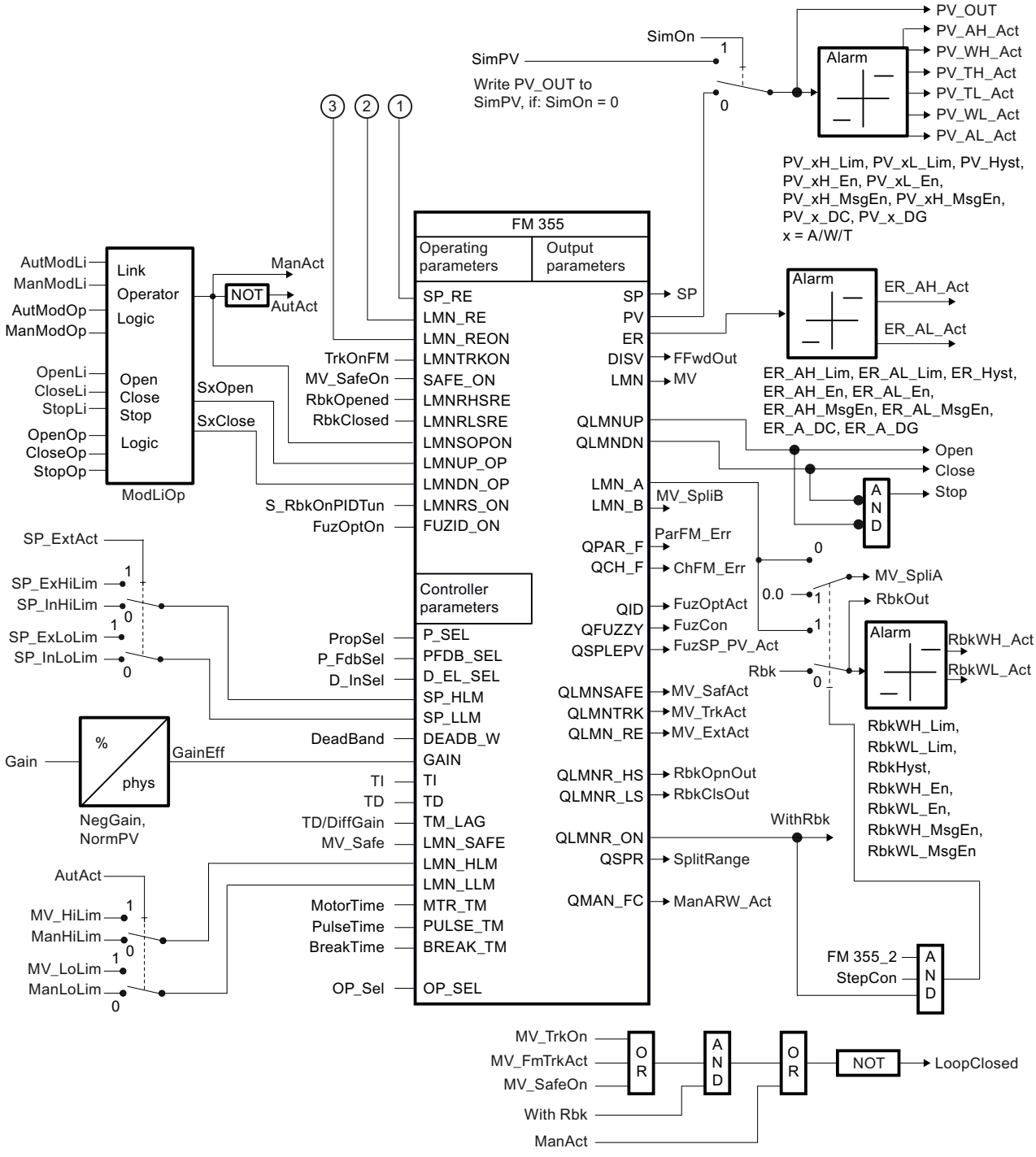
## See also

FmTemp messaging (Page 935)
FmTemp modes (Page 920)
FmTemp block diagram (Page 956)

## 6.4.7 FmTemp block diagram

### FmTemp block diagram






## See also

Description of FmTemp (Page 916)

FmTemp modes (Page 920)

FmTemp functions (Page 922)

FmTemp error handling (Page 934)

FmTemp messaging (Page 935)

FmTemp I/Os (Page 938)

## 6.4.8 Operator control and monitoring

### 6.4.8.1 FmTemp views

#### Views of the FmTemp block

The block FmTemp provides the following views:

- FM controllers standard view (analog) (Page 294)
- FM controllers standard view (pulse controller) (Page 298)
- FM controllers standard view (step controller with position feedback) (Page 302)
- FM controllers standard view (step controller without position feedback) (Page 306)
- Alarm view (Page 336)
- Limit view of FM controllers (Page 322)
- Trend view (Page 341)
- Parameter view of FM controllers (Page 318)
- Preview of FM controllers (Page 331)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icons for PID and FM controller (Page 269)

Refer to the Structure of the faceplate (Page 279) and Block icon structure (Page 260) sections for general information about the faceplate and block icon.

# 6.5 GainSched - Adapting parameter values for a PID controller

## 6.5.1 Description of GainSched

### Object name (type + number) and family

Type + number: FB 1820

Family: Control

### Area of application for GainSched

The block is used for the following applications:

- Continuous adaptation of the parameter values of a PID controller to the current operating point of a non-linear process
- Controller gain
- Integral action time
- Derivative action time

### How it works

If your process requires different PID controller parameters due to its non-linear response at different operating points, you can store optimum parameter sets for up to three different operating points in theGainSched block in the form of a table ("timetable"). The current operating point is represented by a continuously measurable variable X, typically by the actual value of the controller itself. The block ensures that the suitable optimum parameters `Gain`(j), `TI`(j) and `TD`(j) are made available to the controller for each operating point X(j).

If the process is between two operating points, the parameters are calculated by linear interpolation between the optimum values of the two nearest operating points. This allows a bumpless, continuous adaptation of the controller parameters while the process moves from one operating point to another.

The block should be considered a supplementary function for a PID controller to improve the control performance of the PID controller in non-linear processes. The GainSched faceplate is called from the parameter view of the corresponding PID controller using the "Gain scheduler" button.

In contrast to all other function blocks, the GainSched block is implemented as a CFC chart and is generated with the "Compile chart as block type" function. The source chart "`FbGainSchedLim`" is supplied with the library so that you have more options open to you:

- You can use the precompiled function block GainSched from the library if the standard functionality is adequate for your needs.
- If you require special additional functions for gain scheduling in your application (for example more than three operating points, additional logic functions for selecting the parameters), you will need to modify the CFC source chart and compile it as a block type with a different FB number.

If the current value of input parameter X is below the lowest value X1 in the table or above the highest value X3, precisely the controller parameters which are specified at the relevant boundary point X1 or X3 in the table are output.

## Configuration

The GainSched block is placed in the same CFC chart as the assigned controller and interconnected with it as shown in the corresponding template: The output parameters Link2Gain, Link2TI and Link2TD are connected to the inputs Gain, TI and TD of the PID controller. The input X of GainSched is supplied with the measured value for the operating point, typically with the same value as PV of the controller.

You can open the standard view for the GainSched block from the parameter view of a controller (for example PIDConL). Additional information on this topic is available in the section Opening additional faceplates (Page 234).

To specify the parameters for gain scheduling, run separate controller optimizations at each of the intended operating points, for example with a tool such as the PID tuner. Use amplitudes as small as possible to excite the process to capture the approximately linear response in the area of the operating point under investigation. The optimum parameter values calculated by the PID tuner are entered in the relevant row belonging to the operating point in the table of the GainSched block. The table is clearly displayed in the standard view of the faceplate. Make sure that the numeric values are also permanently stored in the data management of the engineering system by reading back the numeric values of the parameters from AS to the ES or entering them manually at the inputs of the CFC block.

For the GainSched block, the Advanced Process Library contains a template for process tag types as an example with an application scenario for this block.

Example of process tag types:

- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 2825)

## Gain scheduling for batch processes

A typical area of application for gain scheduling is in batch processes that, in contrast to continuous processes, cannot be linearized around a fixed operating point because they need to be moved backwards and forwards between different operating points during the course of the batch. Here, there are three application scenarios:

- The controller parameters depend on a single continuously measurable variable that is representative of the operating point, for example, the reactor temperature. This is the normal use case for the GainSched block: The management of the controller parameters is handled in the block and is independent of batch recipes.
- The controller parameters depend on a continuously measurable variable that is representative of the operating point, but there is also a dependency on the materials used in the reaction. Suitable parameter sets for gain scheduling can then be anchored in the recipe and transferred by SIMATIC BATCH to the GainSched block.

- The controller parameters depend only the current phase of the batch. They can then be written directly from the Batch package to the PID controller and no gain scheduling block is necessary. The disadvantage of this is that there is bump in the controller parameters at the transfer from one phase to the next. The controller should be put into manual mode temporarily at the time of the transfer to avoid a bump in the manipulated variable.
- The recipe only specifies which of the controller parameter sets 1 ... 3 is currently required from the GainSched block. The numerical values of the parameter, however, are not anchored in the recipe. In this case input variable `X` of the GainSched block can then be used as the number of the required data record and assigned by the recipe instead of being linked with a measurable process variable. In this case, there are only three discrete values for `X` and the precautions against a change of controller parameters with bump outlined above must be taken because the interpolation abilities of the GainSched block are not used.

In general, it is not necessary to manage the batch parameters (batch number, batch name, etc.) in the GainSched block because the block does not generate any separate messages and there is always a 1-to-1 relationship with a controller block that knows the batch parameters.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

GainSched functions (Page 962)
GainSched messaging (Page 963)
GainSched I/Os (Page 964)
GainSched block diagram (Page 966)
GainSched error handling (Page 963)
GainSched modes (Page 961)

## 6.5.2 GainSched modes

### GainSched modes

The block can be operated using the following modes:

- Automatic mode (Page 79)
- Manual mode (Page 79)

### "Automatic mode"

In "automatic mode" (`ManParOn` = 0) the controller parameters are determined through a polygon in accordance with the settings in the "automatic" area of the parameter view.

You can find general information on "Automatic mode", switching modes and Bumpless switchover in the Manual and automatic mode for control blocks (Page 79) section.

### "Manual mode"

In "manual mode" (`ManParOn` = 1) the controller parameters correspond to the settings in the "manual" area.

You can find general information on "Manual mode", switching modes and bumpless switchover in the section Manual and automatic mode for control blocks (Page 79).

### See also

GainSched block diagram (Page 966)

GainSched I/Os (Page 964)

GainSched messaging (Page 963)

GainSched error handling (Page 963)

GainSched functions (Page 962)

Description of GainSched (Page 959)

## 6.5.3 GainSched functions

### Functions of GainSched

The functions for this block are listed below.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) section. The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 24 | Enabling local operator authorization (Page 169) |

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

**See also**

Description of GainSched (Page 959)
GainSched messaging (Page 963)
GainSched I/Os (Page 964)
GainSched block diagram (Page 966)
GainSched error handling (Page 963)
GainSched modes (Page 961)
GainSched standard view (Page 967)

## 6.5.4 GainSched error handling

**GainSched error handling**

The block does not report any errors.

**See also**

GainSched block diagram (Page 966)
GainSched I/Os (Page 964)
GainSched messaging (Page 963)
GainSched functions (Page 962)
GainSched modes (Page 961)
Description of GainSched (Page 959)

## 6.5.5 GainSched messaging

**Messaging**

This block does not offer messaging.

**See also**

Description of GainSched (Page 959)
GainSched functions (Page 962)
GainSched I/Os (Page 964)
GainSched block diagram (Page 966)

GainSched error handling (Page 963)

GainSched modes (Page 961)

## 6.5.6 GainSched I/Os

### GainSched I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Gain1 | PID gain for operating point 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Gain2 | PID gain for operating point 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Gain3 | PID gain for operating point 3 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| GainOp | PID gain: Input for "manual mode" | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Feature | I/O for additional functions (Page 962) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| ManParOn | 1 = input of PID parameters in "manual mode"<br>0 = use the planned controller parameters from the table | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_In | Input parameter for local operating permission, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| TD_Op | PID derivative action time [s]: Manual input for the operator | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| TD1 | PID derivative action time [s] for operating point 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| TD2 | PID derivative action time [s] for operating point 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| TD3 | PID derivative action time [s] for operating point 3 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| TI_Op | PID integral action time [s]: Manual input for the operator | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| TI1 | PID integral action time [s] for operating point 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| TI2 | PID integral action time [s] for operating point 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| TI3 | PID integral action time [s] for operating point 3 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| X | Process value that defines the operating point | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| X1 | Operating point 1 (support point) for X | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| X2 | Operating point 2 (support point) for X | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| X3 | Operating point 3 (support point) for X | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| X_Unit | Unit of measure for the operating point | INT | 1001 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Link2Gain | Calculated controller gain | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Link2TD | Calculated integral action time | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Link2TI | Calculated derivative action time | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Parameter used to hide operator controls in the faceplate | DWORD | 16#00000000 |

### See also

Description of GainSched (Page 959)
GainSched messaging (Page 963)
GainSched block diagram (Page 966)
GainSched modes (Page 961)

## 6.5.7 GainSched block diagram

### GainSched block diagram

A block diagram is not provided for this block.

### See also

GainSched I/Os (Page 964)
GainSched messaging (Page 963)
GainSched error handling (Page 963)
GainSched functions (Page 962)
GainSched modes (Page 961)
Description of GainSched (Page 959)

## 6.5.8 Operator control and monitoring

### 6.5.8.1 GainSched views

#### Views of the GainSched block

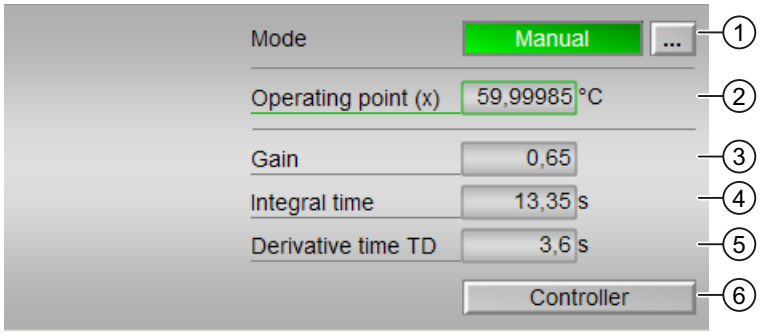
The GainSched block provides the following views:

- GainSched standard view (Page 967)
- GainSched parameter view (Page 968)
- GainSched preview (Page 969)
- Memo view (Page 338)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 6.5.8.2 GainSched standard view

#### GainSched standard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 79)
- Automatic mode (Page 79)

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the operating mode.

In "manual mode", you can specify the values on the OS in the parameter view of this block; they are then output directly via the corresponding output parameters.

In "automatic mode", an interpolation is performed through the interpolation points, which can also be specified in the parameter view.

### (2) Displaying the operating point (X)

Currently used operating point.

### (3) Displaying the gain

The controller gain currently output at the `Link2Gain` output parameter.

### (4) Displaying the integration time `TI`

Integration time currently output at the `Link2TI` output parameter.

### (5) Displaying and changing the derivative time `TD`

Derivative time currently output at the `Link2TD` output parameter.

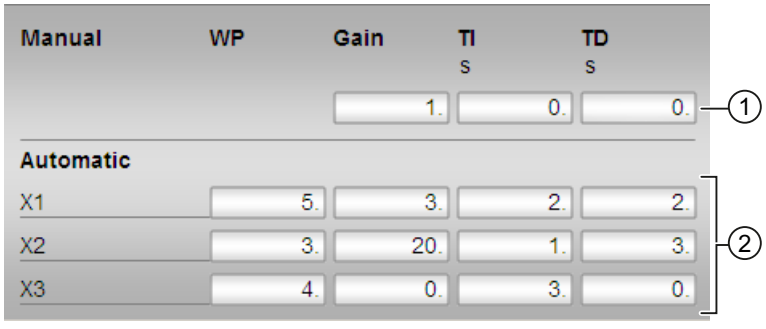
### (6) Navigation button to GainSched block

Use this navigation button to reach the standard view of a controller block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

Additional information is available in the section Opening additional faceplates (Page 234).

## 6.5.8.3 GainSched parameter view

### Parameter view of GainSched




### (1) Displaying and changing the values for the controller parameters in "manual mode"

This is where you enter the values for the parameters to be used in "manual mode" at the corresponding output parameters of the block:

- "Gain": Input parameter `GainOp`
- "TI": Integration time, input parameter `TI_Op`
- "TD": Derivative time, input parameter `TD_Op`

Refer to the section titled Changing values (Page 291) for information on changing the values.

### (2) Displaying and changing the values for the controller parameters in "automatic mode"

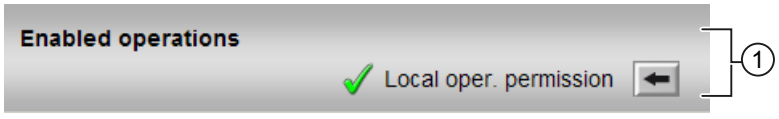
This is where you enter the values for the parameters to be used in "automatic mode" for the interpolation (max. 3 values):

- "X1": Operating point 1, input parameter `X1`
- "X2": Operating point 2, input parameter `X2`
- "X3": Operating point 3, input parameter `X3`
- "Gain": Input parameter `Gain1` ... `Gain3`
- "TI": Integration time, input parameter `TI1` ... `TI3`
- "TD": Derivative time, input parameter `TD1` ... `TD3`

Refer to the section titled Changing values (Page 291) for information on changing the values.

## 6.5.8.4 GainSched preview

### Preview of GainSched




### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

The following enabled operations are shown here:

- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. Additional information is available in the section Operator control permissions (Page 285).

# 6.6 ModPreCon - Model predictive controller

## 6.6.1 Description of ModPreCon

### Object name (type + number) and family

Type + number: FB 1843

Family: Control

### Area of application for ModPreCon

The block is used in much the same way as a PIDConL block for the following applications:

- Fixed setpoint control
- Cascade control
- Ratio control
- Split-range control

In contrast to the PID controllers, this is a multivariable controller.

Compared to Advanced Process Library V7.1.4, the new version provides the following functional enhancements:

- Integrated static operating point optimization,
- Prediction without control response and display of the prediction of the free motion,
- Automatic process trigger for model identification,
- Verification of the sampling time.

### Method of operation and area of application

The block is used for multivariable control (Page 2888) of dynamic processes. It can handle up to four dependent manipulated and controlled variables as well as a measurable disturbance.

In special situations, the ModPreCon block can also be used for particularly difficult dynamic, single-variable controls. It is better than a PID controller, for example, in systems with non-minimum phase (Page 2889) or a strongly oscillating response.

The ModPreCon algorithm only works for stable processes with a step response that settles to a fixed value in a finite time. If the process is unstable in one of the main transfer functions or includes an integrator (level control, for example), the corresponding partial transfer function must be stabilized with a secondary controller.

A simple P controller (proportional component only) is sufficient as a subordinate controller for integrating processes. It is inserted between the manipulated variable output from ModPreCon and the input of the unstable process section, and receives the output of the integrating process section as a manipulated variable. (Unstable data links are stabilized with this approach.)

You can find an explanation of "multivariable controls" and "non-phase-minimum response" in the Help on the Advanced Process Library > Definitions.

## Note on the area of application of the controller: Longer run times

Due to the principle on which they work, the runtime for multivariable controllers is significantly longer than that for PID controllers, because the matrix calculations in the algorithm are much more complex. The runtime is also determined by the number of the process and manipulated variables in the control algorithm. This is why the multivariable controller is unsuited for rapid control and is usually used for slow, complex control tasks.

The computation time required on the CPU is compensated for by the fact that very slow sample times of > 20s are used for the typical ModPreCon applications (see Advanced control templates). The ModPreCon is then typically in OB30 and can be interrupted by faster OBs.

Optimization is called within the ModPreCon block in the program section in which OB1 is executed. In cyclic operation (OB3x), this avoids the additional computing time required by the optimizer, the acyclic time, i.e. which is only relevant for changes to the optimizer inputs. The calculation time load caused by the ModPreCon block is hardy more with optimization than it is for ModPreCon without optimization.

### Operating principle

The ModPreCon block is a model-based predictive multivariable controller. It uses a mathematical model of the process dynamics including all interactions as part of the controller. This model allows the process response to be predicted over a defined period in the future, also known as the prediction horizon.

Based on this prediction, a performance index

$$J = (\vec{w} - \vec{y})^T \cdot R \cdot (\vec{w}-\vec{y}) + \Delta\vec{u}^T \cdot Q \, \Delta\vec{u}$$

is optimized (minimized) where the following applies:

- w contains the time series of the future setpoints,
- y contains the vector of the controlled variables in the future,
- Δu contains the future changes to the manipulated variable.

If you increase the weighting in the Q diagonal matrix, the controller moves its manipulated variables more cautiously resulting in a slower but more robust control response. Using the weighting factors in the R diagonal matrix, you specify the relative significance of the individual controlled variables. A higher weighting (priority) for a controlled variable means that this CV moves more quickly towards the setpoint and remains more accurately at the setpoint in steady state if it is not possible to achieve all setpoints precisely.

The algorithm is a variant of the DMC procedure (**D**ynamic **M**atrix **C**ontrol) in which the optimization problem is solved in the design phase ignoring the constraints. The function block itself contains the analytical solution of the optimization problem. Manipulated variable limitations, both absolute and relating to the gradients, are treated in the algorithm of the function block as hard constraints that must not be violated. This means that precise setpoints or target zones for the controlled variables are taken into account during optimization as well as possible. The target zones for the controlled variables are therefore soft limits, which are maintained as well as possible although they cannot be guaranteed. Using a reference variable filter for future setpoint settings, the control response of the controller can be fine-tuned during operation.

You can achieve significant improvements in control performance when individual disturbances can be measured, for example variations in throughput. In this case, it is a good idea to take into account a model of the influence of this disturbance on the controlled variables when predicting the controlled variables so that the controller can react preemptively to such disturbances.

**Operating point optimization**

The integrated static operating point optimization can be used when at least one controlled variable provides certain degrees of freedom. No exact setpoint is specified for such controlled variables. Instead there is a tolerance band, e.g.

SP2OptHiLim...SP2OptLoLim

within which the process value, CV2, must remain. These areas can be defined for any subset of the relevant controlled variables. From an economic perspective, different values within the tolerance range can be more or less favorable. With the help of the optimization function, the optimal economic point can be found within the tolerance range. This is done by defining a target function (performance criterion), which depends on the manipulated variable and controlled variable of the predictive controller. This can be, for example, the economic yield of plant operation per time unit or it may involve specific costs or energy consumption.

## Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

After installation in CFC, follow the steps outlined below:

1. Excite the process with the controller in manual mode by applying a series of manipulated variable step changes.
2. Record the measured data with the CFC trend display and export it to an archive file.
3. Select the ModPreCon instance in the CFC. Start the MPC Configurator with the command Edit > MPC Configurator. The working steps process identification, controller design and simulation of the closed control loop are executed in the MPC Configurator. You can find a detailed description of this procedure in the MPC Configurator help. You can find the help with a button in the MPC Configurator or directly under ...\Program Files (x86)\SIEMENS \STEP7\S7JMPC\s7jmpctb.chm. (the last letter is the language code).
4. Using the Configurator, create an SCL source code for the user data block (DB). It contains the models and matrices required for a ModPreCon instance.

5. Compile the SCL source code in the engineering system and download it to the AS.
6. Enter the number of the data block at the DB_No input parameter of the ModPreCon block. The values are adopted in the controller by restarting the block using the Restart input parameter.

**Note**

During controller design in the MPC Configurator, a controller cycle time and an OB sampling time are calculated, displayed, and stored in the user data block. You yourself are responsible for the ModPreCon block being called in the cyclic interrupt level suitable for the OB sampling time. This is checked in the current ModPreCon version during initialization. If the SampleTime parameter of the function block does not match the OB_SampleTime parameter of the user data block, a parameter assignment error (ErrorNum=3) is displayed. For controller cycle times greater than 5 s, specify the ModPreCon block in the OB30 and specify the appropriate cycle time for the OB30 in the hardware configuration of the Simatic CPU. Controller cycle times slower than 20 s cannot be set in the hardware configuration. The block would then be called every 20s and the slower sampling time automatically realized by an internal cycle reduction ratio in the block.

For the ModPreCon block, the Advanced Process Library contains a template for a process tag type as examples and there is an example project (where APL_Example_xx, xx designates the language variant) containing various application cases for this block. Several application cases are simulated in the example project and serve to explain how the block works.

Example of a process tag type:

- Model-based predictive control (you can find additional information on this in the Help on Advanced Process Library > PCS 7 Advanced Process Control Templates > Process tag types > Model-based predictive control (ModPreCon))

Application cases in example project:

- Predictive control of a 2x2 multi-variable controlled system (you can find additional information on this in the Help on Advanced Process Library > PCS 7 Advanced Process Control Templates > Example project APL_Example_xx > Predictive control of a 2x2 multi-variable controlled system)
- Predictive control of a non-linear process (you can find additional information on this in the Help on Advanced Process Library > PCS 7 Advanced Process Control Templates > Example project APL_Example_xx > Predictive control of a non-linear process)

## Startup characteristics

When the CPU starts up, the block always starts in manual mode. It is only possible to change to automatic mode when a user data block is loaded and the internal measured value memory in ModPreCon is filled with data.

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

You can find additional information on the `Feature` bit "Set startup characteristics" in the Help on Advanced Process Library > Basics of APL > Selectable block response> Set startup characteristics.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section ModPreCon I/Os (Page 990)

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `AutAct.Value` |
| 6 | Invalid signal status |
| 7 | `ManAct.Value` |
| 8 - 9 | Not used |
| 10 | `MV1TrkOn.Value` AND NOT (`ManAct.Value` OR `OosAct.Value`) |
| 11 | `MV2TrkOn.Value` AND NOT (`ManAct.Value` OR `OosAct.Value`) |
| 12 | `MV3TrkOn.Value` AND NOT (`ManAct.Value` OR `OosAct.Value`) |
| 13 | `MV4TrkOn.Value` AND NOT (`ManAct.Value` OR `OosAct.Value`) |
| 14 | Internal error |
| 15 | `DB_Loaded` |
| 16 | `DV_Model Available` |
| 17 | OptimAct |
| 18 | `NOT(OptimAct)` |
| 19 | `SimOn AND ManAct` |
| 20 | J_Mini |
| 21 | NOT(J_Mini) |
| 22 | ExciteOn AND ManAct.Value |
| 23-30 | Not allocated |
| 31 | `Feature.Bit31`: Display of the predictions in the faceplate |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 - 30 | Not used |
| 31 | `MS_RelOp` |

### See also

ModPreCon functions (Page 976)

ModPreCon messaging (Page 990)

ModPreCon block diagram (Page 1001)

ModPreCon error handling (Page 989)

ModPreCon modes (Page 975)

Model-based predictive control (ModPreCon) (Page 2849)

Predictive control of a 2x2 multi-variable controlled system (ModPreConSim) (Page 2879)

Predictive control of a non-linear process (ModPreConNonLinSim) (Page 2879)

## 6.6.2 ModPreCon modes

### Operating modes of ModPreCon

The block can be operated using the following modes:

- Automatic mode (Page 79)
- Manual mode (Page 79)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

The aforementioned operating modes are valid for the block with all control channels (`MV1` ... `MV4`). Moreover, individual control channels can be tracked; see chapter ModPreCon functions (Page 976) for more information.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and bumpless switchover in the Manual and automatic mode for control blocks (Page 79) section.

**Note**

In contrast to PID controllers, it is permitted to run the ModPreCon block in "automatic mode" without its actuating signals affecting the process because there is no risk of integrator windup.

### "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the Manual and automatic mode for control blocks (Page 79) section.

### "Out of service"

You can find general information about the "Out of service" mode in the Out of service (Page 77) section.

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit5` =1 and `Feature.Bit5` =0 is as shown below:

| Output Parameters | `Feature.Bit5` =0 | `Feature.Bit5` =1 |
|---|---|---|
| SP1Out.ST | 16#XX | 16#60 |
| SP2Out.ST | 16#XX | 16#60 |

| Output Parameters | `Feature.Bit5 =0` | `Feature.Bit5 =1` |
|---|---|---|
| SP3Out.ST | 16#XX | 16#60 |
| SP4Out.ST | 16#XX | 16#60 |
| SP1OpOut.ST | 16#XX | 16#60 |
| SP2OpOut.ST | 16#XX | 16#60 |
| SP3OpOut.ST | 16#XX | 16#60 |
| SP4OpOut.ST | 16#XX | 16#60 |
| CV1Out.ST | 16#80 | 16#60 |
| CV2Out.ST | 16#80 | 16#60 |
| CV3Out.ST | 16#80 | 16#60 |
| CV4Out.ST | 16#80 | 16#60 |

**See also**

ModPreCon block diagram (Page 1001)
ModPreCon I/Os (Page 990)
ModPreCon messaging (Page 990)
ModPreCon error handling (Page 989)
Description of ModPreCon (Page 970)

## 6.6.3 ModPreCon functions

### Functions of ModPreCon

The functions for this block are listed below.

### Generating and limiting the manipulated variable

The manipulated variable `MV1` ... `MV4` (hereinafter referred to as `MVx`, x = 1 ... 4) can be generated as follows:

| ManAct | MVxTrkOn | MVx | Limit monitoring | State |
|---|---|---|---|---|
| 1 | - | `Manx` | `ManxHiLim`<br>`ManxLoLim` | "Manual mode", set by the operator |
| 0 | 1 | `MVxTrk` | `MVxHiLim`<br>`MVxLoLim` | Tracking with limitation |
| 0 | 0 | Automatic manipulated variable | `MVxHiLim`<br>`MVxLoLim` | "Automatic mode": Predictive controller algorithm |

Remark

If the controller is in "Out of service" mode, the output parameters `MV1` ... `MV4`, depending on the `Feature` Bit (Neutral position manipulated variable takes effect at startup (Page 176)), are set to the last valid value in manual mode or the corresponding neutral position manipulated

variable (SafePos1 ... SafePos4). Refer to the Out of service (Page 77) section for more on this.

The limited operating range (between MVxHiLim and MVxLoLim) is typically smaller in automatic mode than in manual mode. With regard to the limited range of validity of a linear process model for approximating a non-linear process response, this allows the stability of the closed control loop to be guaranteed within the control range in automatic mode.

The gradients of the manipulated variable (changes per second) are limited to MV1RaLim to MV4RaLim in "automatic mode". Gradient limitation applies both to the positive and negative directions.

### Tracking and limiting a manipulated variable

The block provides the standard function Tracking and limiting a manipulated variable (Page 222).

In contrast to PID controllers, tracking the manipulated variables (MV1 ... MV4) is enabled for specific channels via one of the input parameters MV1TrkOn ... MV4TrkOn. The corresponding manipulated variable is then tracked by the interconnectable input parameters MV1Trk ... MV4Trk.

### Setting the setpoint internally

With this block, the setpoint must always be set internally at the I/Os SP1 ... SP4. These are normally operated in the faceplate. In special situations, you can interconnect the setpoints but they can then no longer be changed using the faceplate.

### Setpoint tracking in manual mode

In this situation (SP_TrackCV = 1), the internal setpoints SP1 ... SP4 are tracked to the assigned process values CV1 ... CV4 in "manual mode". This function allows a bumpless transfer to "automatic mode". After the transfer, the setpoints can be changed by the operator again.

### Setpoint filters

The setpoint filter is the only way of changing the action of the predictive controller without having to create a new user data block with the MPC Configurator and reinitialize the controller. The specified time constant PreFilt1 ... PreFilt4 of the setpoint filter can be interpreted as the required settling time of this CV channel following a setpoint step change. As the time constant setting increases, the controller works more slowly and less aggressively. In particular, this reduces the influence of a setpoint step change in one control channel on neighboring control channels.

Internally, the ModPreCon block works with sets of future setpoints that are compared with the predicted movements of the controlled variables. Without the setpoint filter, it is assumed that the current setpoint will continue to remain valid in the future within the prediction horizon. If there is a setpoint step change, this means that the full value of the new setpoint will be required in the near future although the process cannot achieve this (according to the prediction). With the setpoint filter, an asymptotic setpoint trajectory (first order) is calculated from the current

process value to the required setpoint so that the required setpoint is reached in the specified time.

**Note**

The setpoint filter also comes into effect without a setpoint step if the process value deviates significantly from the setpoint due to disturbances. This means that the filter not only slows down the control response but indirectly also the response to disturbances.

The control response can only be slowed down by the setpoint filter and not accelerated; when the value is 0, the prefilter is deactivated. It is therefore advisable to set the basic controller action in the MPC Configurator with the "Manipulated variable change penalty" parameter and then to optimize this in the software using the function for simulation of the closed control loop. The setpoint filter should then only be used for fine tuning of the action in the operational system.

**Note**

Do not select an excessively long filter time constant. Values that are greater than the control horizon of the predictive controller (s. ModPreCon user DB, header comment of the SCL source: "control horizon" times "controller sample time" in seconds) can have undesirable effects when the dead band is used at the same time. If you wish to dramatically slow down the reaction of the controller, it is better to use the penalty of the manipulated variable changes in the MPC Configurator.

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Controlled variable (`SimCVx` , `SimCVxLi`)

### Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 240).

### Error signal generation and dead band

The block provides the standard function Error signal generation and dead band (Page 217).

In the predictive controller, the error signal is generated over the entire prediction horizon for each control channel as the deviation between the predicted movement of the process (starting at the current process value `CV1` ... `CV4`) and future setpoint settings (ending at `SP1` ... `SP4`) and used to calculate the manipulated variable.

In principle, the effect of the dead bands `SP1DeadBand` to `SP4Deadband` is the same as in a PID controller, but extends over the entire future prediction horizon. In other words, if, for example, the predicted controlled variable `CV1` in the entire prediction horizon is within the band `SP1` ± `SP1DeadBand`, the controller sees no reason whatsoever to change any manipulated variable. These are therefore also known as `CV` bands. In contrast to the manipulated variable limits, these are not hard constraints that need to be adhered to at all costs.

In multivariable controllers, it is advisable to make use of the fact that from the perspective of the application only some of the controlled variables need to move to a specified setpoint exactly while others only need to remain within a defined range.

A typical example would be quality characteristics for which a tolerance band is specified. While a dead band in a PID controller tends to put stability at risk, CV bands in individual controller channels generally relieve the multivariable controller overall.

Using CV bands, the action of a soft override control can be achieved.

### Use case for error signal generation with dead band

As long as the pressure in a reactor remains within the set safety limits, the controller is interested only in product quality. However, as soon as the pressure threatens to leave the permitted range (in other words, in the prediction it moves towards an illegal value in the future), the pressure control cuts in. By weighting the controlled variables in the performance index (see MPC Configurator), the user can specify that predicted violations of the pressure limits are given a particularly high weighting.

### Predictive controller algorithm

The ModPreCon block is derived from the familiar DMC algorithm (**D**ynamic **M**atrix **C**ontrol) . Future changes to the manipulated variable within the control horizon are calculated according to the formula:

$$\Delta\underline{\vec{u}} = \underline{C} \cdot (\underline{\vec{w}} - \underline{\vec{f}})$$

Where:

- w contains the time series of the future setpoints
- f contains the predicted free movement of the controlled variables (with constant manipulated variables) in the future
- C is the constant controller matrix calculated by the MPC Configurator. C includes both the process model and the weighting of the manipulated variable changes and the controlled variables from the performance index of the optimization.

Based on the principle of the receding horizon, only the first value is taken from the vector of the optimum manipulated variable changes over the entire control horizon and applied to the process. In the next step, the newly arrived process values are taken into account and the calculation repeated over the entire prediction horizon.

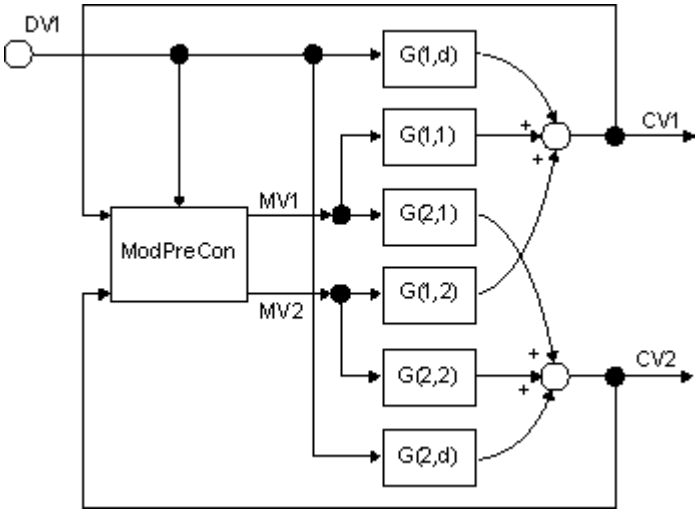
With predictive controllers, the manipulated variable changes are based on the control deviations predicted in the future, while with a PID controller, they are based on error signal of the past (possibly also integrated). This can be interpreted as a "looking ahead" strategy.

### Anti-windup

When manipulated variable limits are active, anti-windup measures are taken automatically within the controller. The prediction equations use the real limited values of the manipulated variable instead of theoretically calculated values.

### Model-based disturbance compensation

Model-based disturbance compensation can and should be used when a known disturbance has a strong influence on the process and its cause can be measured.




The effects of a measurable disturbance (`DV1` I/O) on all controlled variables `CV1` ... `CV4` can be estimated when the controller is taken into manual mode. This means that no movements of the controlled variables whatsoever result from changes to the manipulated variable and all movements result from the disturbance. If the disturbance can be measured but cannot be actively adjusted, it may be necessary to search through a data archive to find the time segments in which the disturbance changed.

The identification of the transfer functions from the disturbance variable `DV1` to all controlled variables `CV1` ... `CV4` (disturbance model, in the graphic above G(1.d) and G(2.d)) is performed with the MPC Configurator and is analogous to the identification of the main transfer functions (G(1.1) to G(2.2)). The measured disturbance variable is then switched to the `DV1` input of the ModPreCon block and disturbance compensation is activated with `DV_On` = 1. As a result, the effect of the measurable disturbance is taken into account in the prediction and the controller can start counter measures in advance before the disturbance can have a massive influence on the controlled variables.

Such disturbance compensation is especially effective when the disturbance is constant in sections and changes from time to time. If a disturbance changes constantly or oscillates, however, the feedforward control is not enabled during operation of the controller to avoid constant oscillation of the manipulated variables, although it should be taken into account in the MPC Configurator when creating the process model.

If there is no disturbance model in the user data block, the `DV1` input is ignored.

Typical examples of measurable disturbances are inlet volumes in distillation columns or throughput of continuous reactors.

### Predictive controller with more than one measurable disturbance

If you want to plan for more than one measurable disturbance in an application, but do not need all four disturbances from the ModPreCon block, you can dedicate the first of the previously unused control channels for disturbance variable feedforwarding.

Example: You only have two control actions available, so only use MV1 and MV2. Then connect the additional measurable disturbance variable DV2 to the MV3Trk input parameter and set MV3TrkOn = 1. For recording training data for the predictive controller, declare MV3Trk as the third disturbance and also use the CFC trend recorder to record the effect of changes to DV2 on all controlled variables.

Use the MPC Configurator to then determine a process model that describes the effects of DV2. However, if the DV2 in "automatic mode" of the controller changes due to external influences, the effect of such change is taken into consideration for predicting future process reactions through MV3Trk, and predictive methods can be used to compensate. The performance of the disturbance compensation is exactly the same as that for regular feedforward control via the input parameters DV and DV_On = 1.

If you want to disable this disturbance compensation in runtime with the unused MV3 manipulated variable, you need to insert a MV3Trk a selector block in front of the SelA02In input. This allows you to set a constant zero for MV3Trk instead of the measured value DV2, which stops the effect of MV3 on prediction. (Due to this reassignment, MV3TrkOn must always remain 1 to prevent the controller from changing the value of MV3.)

This way, up to four measurable disturbances can be selected. However, the sum of the manipulated variables and disturbances may not exceed a total number of five.

### Control of square and non-square systems

In multivariable controllers, the number of manipulated variables should ideally be the same as the number of controlled variables. This is known as a "square system". As long as constraints do not influence operation, the controller can, in principle, track all controlled variables exactly to the selected setpoints.

If there are less manipulated variables than controlled variables, or individual manipulated variables have reached their limits, there is no freedom in the control problem. This means that it is not possible for all setpoints to be reached exactly.

The ModPreCon algorithm then finds a compromise that can be influenced by the selection of controlled variable weights (priorities) in the MPC Configurator: Controlled variables with higher priority will have lower control deviations.

**Note**

Since the ModPreCon block is a lean predictive controller algorithm without online optimization, there can be no general guarantee that the compromise found is optimum in a mathematical sense; in other words, it is the minimum of the fit function taking into account the manipulated variable limits. In most practical situations, however, the controller finds sensible compromises.

The static operating point optimization it not a dynamic online optimization in this sense, i.e. it does not change anything in the above-mentioned restriction.

If there are more manipulated variables than controlled variables or if some of the controlled variables are already within their setpoint bands, there is surplus freedom in the control problem. A lean predictive controller algorithm, however, cannot recognize this situation explicitly and use the free manipulated variables for optimization. The ModPreCon block therefore moves all manipulated variables to values that meet the aims in terms of controlled variables and then leaves them there. In some situations, however, it can be useful to provide the

controller with more manipulated variables than controlled variables, for example when the effect of individual manipulated variables is too restricted.

Another approach is to define the excess manipulated variables as pseudo controlled variables at the same time. You do this by assigning a setpoint with low priority to the pseudo controlled variables. The controller then attempts to achieve the important control aims as first priority and, at the same time, attempts to reach certain ideal values for the individual manipulated variables.

## Control of linear and non-linear systems

The ModPreCon algorithm is based on a linear, time invariant process model. As a result, in much the same way as a PID controller, it is suitable above all for controlling non-linear systems around a fixed operating point.

Again analogous to the PID controller, there are, however, several possibilities with which the area of application can be extended with non-linear systems:

**Compensation functions between controller and controlled system:**

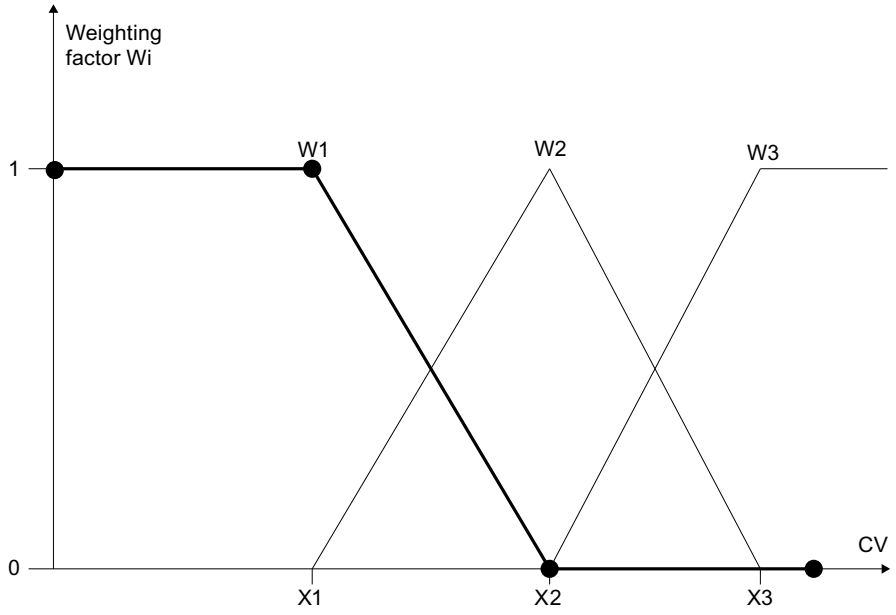
It is, for example, possible to compensate the effect of a non-linear valve characteristic curve using a polygon block between the `MV` output and the control input of the valve block. Care must be taken when implementing the manipulated variable limits. In the same way, the effect of a non-linearity at the output of the controlled system (for example a sensor characteristic curve) can be compensated by a polygon block before the `CV` input of the controller. Remember that the corresponding SP must also be transformed accordingly. In both cases, the compensation functions become part of the controlled system from the perspective of the controller. The aim is always to keep the overall response of the controlled system consisting of process and compensation elements as linear as possible.

**Multimodel control:**

This approach is related to the basic idea of operating-point-based parameter control with PID controllers. Since the model parameter of the ModPreCon block cannot be modified in runtime, however, the control strategy for selecting the suitable parameter set becomes a control strategy for selecting the suitable model.

Several ModPreCon instances with different models for different operating points run at the same time. The local optimal models are determined by starting the process at the various operating points with small amplitudes, so that only the reaction of the non-linear process in the ambience of this operating point is registered.

The final manipulated variable for each manipulated variable is formed as a weighted mean value of the manipulated variables proposed by the controller instances. (It is recommended that experiments for starting the process for the MPC Configurator are only performed after implementing the functions for adding the manipulated variables, in order to ensure that the same conditions applicable when the model is in operation actually take effect.)




The weighting factors 0 ... 1 are formed in the same way as the membership functions known from fuzzy logic so that the sum of all weights is always one and each controller has the highest weighting at its own operating point. A polygon with 4 or 5 interpolation points is used to calculate each individual weighting factor. The weighting factors are calculated based on a specific measurable `PV` variable of the process, which is representative for the operating point of the process. This can be one of the `CVx` controlled variables, although it does not have to be. The abscissa of the interpolation points of all polygons is selected in such a way that they cover the entire value range of `PV` in order to avoid extrapolation errors.

One should note in this regard that the only non-linear effects in the full multi-variable control loop that can be modeled are those that correlate exactly to a representative `PV` variable. This approach is therefore not suitable for cases in which individual partial transfer functions demonstrate non-linear effects that depend on various, totally independent variables.

To ensure the stability of the overall control loop, all subcontrollers must be at least stable at all operating points. In contrast to PID controllers however, an MPC is not affected by windup problems if it temporarily runs in "automatic mode" but cannot intervene in the real process (weighting factor zero).

One of the controller instances is defined as the main controller and shown in the operator faceplate on the OS. All others are connected in such a way that they adopt the operating mode ("manual"/"automatic mode") and the setpoints from the main controller. The manual manipulated variables are passed to the secondary controller via the tracking inputs. This means that no operator intervention is required on secondary controllers.

**Note**

The manipulated variables of the main controller can be used when switching from automatic to manual mode. If the process was in the operating range of a secondary controller beforehand, the current manipulated variables in the process can deviate significantly. In this case, you should apply the actual manipulated values in effect by manual input in the faceplate of the primary controller.

You can find an example for multi-model controlling in the example project of Advanced Process Library under Predictive control of a non-linear process (ModPreConNonLinSim) (Page 2879).

**Trajectory control:**

This approach neatly combines the advantages of an open loop controller (Feedforward Control) with those of a closed loop controller with process value feedback (Closed Loop Control). The controller follows a previously optimized trajectory of setpoints and manipulated variables; in other words, it only needs to compensate small deviations between the stored trajectory and the current plant state. A trajectory is an optimum series of manipulated variables over time and the process values that match them. The required manipulated variables are read via the inputs `MV1Traj` ... `MV4Traj` into the ModPreCon block and added to the values of the manipulated variable calculated by the algorithm (in automatic mode only). Among other things, the advantage of this is that the effective manipulated variable acting on the process can be configured and is limited to the sum of the trajectory and controller action. The process values from the trajectory are switched to the corresponding setpoint specifications `SP1` ... `SP4` of the controller. As long as the process reacts exactly as planned in the trajectory, it will respond to the series of manipulated variables from the trajectory with the corresponding series of process values and the control deviation is zero. It is generally known that a non-linear dynamic process can linearized around a fixed operating point or a steady state of the system. It is also possible to linearize it around a trajectory.

## Forming the signal status for blocks

The block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `DV1.ST`
- `CV1.ST`
- `CV2.ST`
- `CV3.ST`
- `CV4.ST`

## Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 4 | Setting switch or button mode (Page 177) |
| 5 | Changes signal status of outputs in OOS (Page 208) |
| 8 | Delete history of MV and DV moves during restart (Page 201) |
| 15 | Neutral position manipulated variable takes effect with "out of service" operating mode (Page 177) |

| Bit | Function |
|---|---|
| 16 | Neutral position manipulated variable takes effect at startup (Page 176) |
| 24 | Enabling local operator authorization (Page 169) |

## Operator permissions

The block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" |
| 1 | Not used |
| 2 | 1 = Operator can switch to "Out of service" mode |
| 3 | Not used |
| 4 | Not used |
| 5 | 1 = Operator can change the setpoint 1 |
| 6 | 1 = Operator can change the manipulated variables of all channels |
| 7 | 1 = Operator can change operating high limits of the setpoints for all channels |
| 8 | 1 = Operator can change operating low limits of the setpoints for all channels |
| 9 | 1 = Operator can change the setpoint 2 |
| 10 | 1 = Operator can change the setpoint 3 |
| 11 | 1 = Operator can change the setpoint 4 |
| 12 | 1 = Operator can change the setpoint filter of all channels |
| 13 - 16 | Not used |
| 17 | 1 = Operator can enable the track setpoint in "manual mode" function |
| 18 | 1 = Operator can activate the model-based disturbance compensation function |
| 19 | 1 = Operator can enable the function "Prediction without control response" |
| 20 - 22 | Not used |
| 23 | 1 = Operator can change the dead band parameter of all channels |
| 24 | 1 = Operator can change the tuning parameters |
| 25 | 1 = Operator can change the simulation value SimPV1..4 |
| 26 | 1 = Operator can activate the Simulation function |
| 27 | 1 = Operator can activate the Release for maintenance function |
| 28 | 1 = Operator can change the manipulated variable limits of all channels |
| 29 | 1 = Operator can change the gradient limits of manipulated variables of all channels |
| 30 | 0 = Operator can set the integrated static operating point optimization |
| 31 | 1 = Operator can set the integrated static operating point optimization |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Release for maintenance

The block provides the standard function Release for maintenance (Page 70).

### Specifying the display area for process and setpoint values as well as operations

This block provides the standard function Display and operator input area for process values and setpoints (Page 233).

In contrast to PID controllers, there are no separate parameters for bar limits. The setpoints limits are used for all setpoint and actual value bars; manual limits are used as bar limits for all manipulated variable bars.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Integrated static operating point optimization

The integrated static operating point optimization can be used if no exact SPi setpoint is specified for at least one controlled variable (index i =1...4), but rather the CVi process value must remain within a tolerance range SPiOptHiLim...SPiOptLoLim. The tolerance range must, of course, be within the valid setpoint limits SPiHiLim...SPiLoLim for this control channel. The tolerance range SPiOptHiLim...SPiOptLoLim is not coupled to the operator-controlled setpoint SPi. If the setpoint is changed, the tolerance range is not automatically shifted. If you want to do this nevertheless, interconnect the SPiOpOut output parameters via two adders to the width of the tolerance range at the inputs SPiOptHiLim and SPiOptLoLim.

From an economic perspective, different values within the tolerance range can be more or less favorable. With the help of the optimization function, the optimal economic point can be found within the tolerance range.

This is done by defining a target function (performance criterion), which depends on the manipulated variable and controlled variable of the predictive controller. This can be, for example, the economic yield of plant operation per time unit or it may involve specific costs or energy consumption.

J= GradMV1*MV1 +GradMV2*MV2 +GradMV3*MV3 +GradMV4*MV4

+GradCV1*CV1 +GradCV2*CV2 +GradCV3*CV3 +GradCV4*CV4

+J0

You specify the individual GradXVi coefficients of the gradient vector as input variables at the ModPreCon function block in the CFC or in the parameter view of the faceplate. If individual coefficients vary with time, e.g. they are dependent on current market prices, you can also interconnect these input variables. If individual manipulated or controlled variables have no influence on the performance criterion, leave the corresponding coefficients at the default value, zero.

You can use the binary J_Mini input parameter to specify whether the target function is to be maximized or minimized, based on whether this involves yields or costs (J_Mini = 1 : minimization).

The term J0 combines all contributions to the target function that do not depend on manipulated variables and controlled variables. These contributions have no effect on the optimum values in the decision variables, but are applied for calculating the current value of employed performance criterion similar to the above-mentioned formula.

Within the controller, the terms of the target function that depend on manipulated variables are converted to make their dependence on the controlled variables visible. To do this, the inverse stationary process model from the MPC Configurator is used. This requires that the number of manipulated variables matches the number of controlled variables. If the number of manipulated variables does not match the number of controlled variables, the largest possible square submodel in the matrix of the transfer functions is truncated from the top left. If there are more manipulated variables than controlled variables, for example, only the first manipulated variables are used, in accordance with the number of controlled variables.

Constraints for the controlled variables take the form of the above-mentioned tolerance ranges for setpoints. The controller takes care of adhering to the manipulated variable limits, in any case; they do not have to separately specified as constraints for the optimization.

Enable the optimization using the binary input variable OptimizeOn in the controller faceplate. The optimizer then returns setpoints within the tolerance ranges that are optimal for the performance criterion. These setpoints are then sent to the control algorithm, which handles them in the same way as conventionally specified setpoints (with or without dead band). The operable SP1...SP4 setpoints are not tracked to the optimized setpoints; when optimization is disabled, the old setpoints from the faceplate take effect once again. When selecting variables for archiving and graphic plotter, ensure you use the SP1Out...SP4Out setpoint actually in effect and not the SP1...SP4. input variables.

The current value of the performance criterion is displayed at the J_Actual output variables.

You can find additional information about the topic of static operating point optimization in the online help for the MPC configuration editor.

### Display of the prediction of free movement

The prediction of free motion is a forecast for the future behavior of the process within the overall prediction horizon, under the assumption that all manipulated variables are frozen at their current values. The time length of the prediction horizon is indicated in the output parameter PrediHorizon in the [s] unit.

The prediction of free motion is recalculated in each sampling step within the control algorithm. If the manipulated variables is to a constant value in manual mode, the prediction of the free movement is actually a realistic prediction for the future process response. It can therefore be represented graphically in the faceplate at least in terms of quality. For this purpose, five equidistant interpolation values are copied from the prediction horizon, and displayed in the standard view of the faceplate as a vertical bar next to the current process value.

Example: The prediction horizon is 1800s=30min and the current time is labeled with the index k. The prediction for k+6min, and next to it k+12min, up to k+30min then appears on the right next to the bar of the current process value. If the upper border of the bar is conceptually connected with a line (red in the picture red), you can imaging the curve created by the future course of the process value over the next half hour.

In automatic mode (tracking function is switched off), the value of the manipulated variables changes with each sampling step. The prediction of free movement is then only a fictional mathematical formulation within the algorithm, and not a realistic prediction for the future process response. This is why the prediction is only displayed in manual mode or in automatic mode with active tracking function. The display can be generally suppressed using `Feature.Bit31`.

### Prediction without control response

In this special "operating mode" (comparable to the block-internal simulation), the controller only monitors the process and indicates what it would like to do in the next sampling step without actively intervening in the process. This allows you as the user to build trust before "switching active" the controller the first time, i.e. actively intervening in the process.

"Prediction Mode" is activated via the binary input variable PredictMode or in "Parameters" in the faceplate view. Setpoints and process values are read as in normal automatic mode. The prediction of the free movement and manipulated variable change for the next sampling step are calculated as in normal automatic mode. The starting point for the prediction of the manipulated variable for the next sampling step, however, is the current process value of the follow-up control loop at the MV1Trk...MV4Trk tracking inputs. The predicted manipulated variables are not output at the normal outputs MV1...4, but rather at the MV1Pred...MV4Pred outputs, which were especially introduced for this purpose, and are displayed in the standard view of the faceplate on the left next to MV1...4, as long as "Prediction Mode" is active.

- When the controller is in automatic mode, in "Prediction Mode" all MVi (i=1..4) are set to match the assigned MViTrk input parameters, similar to tracking mode.
- When the controller in manual mode, all MVi are set to the desired manual values regardless of "Prediction Mode".
- When "Prediction Mode" is disabled, all MViPred always equal the assigned MVi.

### Automatic process trigger for model identification

In order to determine the process model for the model predictive controller, the process must be artificially triggered in order to observe its dynamic response and record it in the form of training data. This trigger can be specified manually in manual mode of the controller.

Alternatively, a suitable trigger signal can be generated automatically in the form a defined, symmetrical sequence of manipulated variable jumps. The trigger signals are calculated by an auxiliary function block, "AutoExcitation", which is built into the process tag type and interconnected with ModPreCon .

Additional MV1Excite...MV4Excite input variables are required for this on the controller. Process triggering performed in manual mode of the controller, because automatic mode cannot be activated before modeling. The new process trigger "operating mode" can only be controlled via the ExciteOn input bit on the Engineering System, and not on the operator station, since the CFC is needed for data recording in any case. However, process triggering must be displayed on the OS in standard view at the lower left.

Manual intervention per faceplate remains possible even during the triggering. The values of the MV1Excite...MV4Excite input parameters are therefore only written to the MV1Man...MV4Man manual values event-based, but only if they change.

You can find additional details on automatic process triggering in the online help for the MPC configuration editor.

### See also

Description of ModPreCon (Page 970)

ModPreCon messaging (Page 990)

ModPreCon I/Os (Page 990)

ModPreCon block diagram (Page 1001)

ModPreCon error handling (Page 989)

ModPreCon modes (Page 975)

## 6.6.4 ModPreCon error handling

### Error handling of ModPreCon

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` output parameter can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Default value when the block is installed; this message is irrelevant. |
| 0 | There is no error. |
| 3 | Configuration error: Block is inserted in the wrong OB, cycle time of the OB is not suited for controller configuration. |
| 32 | The value of `CV1` can no longer be displayed in the real number field or is not a number. |
| 33 | The value of `CV2` can no longer be displayed in the real number field or is not a number. |
| 34 | The value of `CV3` can no longer be displayed in the real number field or is not a number. |
| 35 | The value of `CV4` can no longer be displayed in the real number field or is not a number. |
| 36 | The `MV_Trk1` value can no longer be displayed in the real number field or is not a number. |
| 37 | The `MV_Trk2` value can no longer be displayed in the real number field or is not a number. |
| 38 | The `MV_Trk3` value can no longer be displayed in the real number field or is not a number. |
| 39 | The `MV_Trk4` value can no longer be displayed in the real number field or is not a number. |

| Error number | Meaning of the error number |
|---|---|
| 51 | `AutModLi` = 1 and `ManModLi` = 1 |
| 90 | The controller matrix could not be loaded from the user data block. |

The ErrorOpt output parameter is used to output the status of the lower-level LPOptim block. See ModPreCon I/Os (Page 990) for more.

**See also**

ModPreCon block diagram (Page 1001)

ModPreCon messaging (Page 990)

ModPreCon functions (Page 976)

ModPreCon modes (Page 975)

Description of ModPreCon (Page 970)

Description of LPOptim (Page 1229)

## 6.6.5 ModPreCon messaging

**Messaging**

This block does not offer messaging.

**See also**

Description of ModPreCon (Page 970)

ModPreCon functions (Page 976)

ModPreCon I/Os (Page 990)

ModPreCon block diagram (Page 1001)

ModPreCon error handling (Page 989)

ModPreCon modes (Page 975)

## 6.6.6 ModPreCon I/Os

I/Os of ModPreCon

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

## Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutModLi* | 1 = Automatic mode via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutModOp* | 1 = Automatic mode via operator (controlled by ModLiOp = 0) | BOOL | 0 |
| BatchEn | 1 = Enabled for allocation by batch control | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | STRING[32] | '' |
| BlkTypeInfo* | Block Type Version and FB Number of block-type<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| CV1 | Control variable 1 (process value) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| CV1_Unit | Unit of measure for control variable 1 (process value) | INT | 1001 |
| CV2 | Manipulated variable 2 (process value) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| CV2_Unit | Unit of measure for manipulated variable 2 (process value) | INT | 1001 |
| CV3 | Manipulated variable 3 (process value) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| CV3_Unit | Unit of measure for manipulated variable 3 (process value) | INT | 1001 |
| CV4 | manipulated variable 4 (process value) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| CV4_Unit | Unit of measure for manipulated variable 4 (process value) | INT | 1001 |
| DB_No | Number of the data block in which the controller data is saved. | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| DV_On | 1 = Activate the feedforward control from DV1 | BOOL | 1 |
| DV1 | Disturbance variable | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| ExciteOn | 1 = Automatic process trigger; MViExcite input parameters are written to the MVi outputs | BOOL | 0 |
| Feature | I/O for additional ModPreCon functions (Page 976) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| GradCV1 | Gradient vector for performance criterion, element (factor) for CV1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| GradCV2 | Gradient vector for performance criterion, element (factor) for CV2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| GradCV3 | Gradient vector for performance criterion, element (factor) for CV3 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| GradCV3 | Gradient vector for performance criterion, element (factor) for CV3 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| GradCV4 | Gradient vector for performance criterion, element (factor) for CV4 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| GradMV1 | Gradient vector for performance criterion, element (factor) for MV1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| GradMV2 | Gradient vector for performance criterion, element (factor) for MV2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| GradMV3 | Gradient vector for performance criterion, element (factor) for MV3 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| GradMV4 | Gradient vector for performance criterion, element (factor) for MV4 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| J_Actual_Unit | Physical unit of performance J_Actual | INT | 0 |
| J_Mini | 1 = Minimize, 0 = maximize | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| J0 | Value of the performance criterion in the operating point | REAL | 0 |
| ManModLi* | 1 = Manual mode via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManModOp* | 1 = Manual mode via OS operator (controlled by ModLiOp = 0) | BOOL | 0 |
| ModLiOp | Operating mode switchover by:<br>• 0 = Operator<br>• 1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_RelOp* | 1 = Release for maintenance by OS operator | BOOL | 0 |
| MV1_Unit | Unit of measure for manipulated variable 1 | INT | 1342 |
| MV1Excite | MV1 for automatic process trigger | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV1HiLim | High limit of manipulated variable MV1 | REAL | 100.0 |
| MV1LoLim | Low limit of manipulated variable MV1 | REAL | • 0.0 |
| MV1Man* | Manual value: Operator input for setting the manipulated variable MV1 in manual mode | REAL | 0.0 |
| MV1ManHiLim | High limit of manipulated variable MV1 in manual mode | REAL | 100.0 |
| MV1ManLoLim | Low limit of manipulated variable MV1 in manual mode | REAL | 0.0 |
| MV1RaLim | Gradient limit of the manipulated variable MV1 per sampling step | REAL | 100.0 |
| MV1Traj | Trajectory value that is added to the manipulated variable MV1 | REAL | 0.0 |
| MV1Trk | Tracking value for the manipulated variable MV1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV1TrkOn | 1 = Tracking of manipulated variable MV1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV2_Unit | Unit of measure for manipulated variable 2 | INT | 1342 |
| MV2Excite | MV2 for automatic process trigger | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV2HiLim | High limit of manipulated variable MV2 | REAL | 100.0 |
| MV2LoLim | Low limit of manipulated variable MV2 | REAL | 0.0 |
| MV2Man* | Manual value: Operator input for setting the manipulated variable MV2 in manual mode | REAL | 0.0 |
| MV2ManHiLim | High limit of manipulated variable MV2 in manual mode | REAL | 100.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MV2ManLoLim | Low limit of manipulated variable MV2 in manual mode | REAL | 0.0 |
| MV2RaLim | Gradient limit of the manipulated variable MV2 per sampling step | REAL | 100.0 |
| MV2Traj | Trajectory value that is added to the manipulated variable MV2 | REAL | 0.0 |
| MV2Trk | Tracking value for the manipulated variable MV2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV2TrkOn | 1 = Tracking of manipulated variable MV2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV3_Unit | Unit of measure for manipulated variable 3 | INT | 1342 |
| MV3Excite | MV3 for automatic process trigger | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV3HiLim | High limit of manipulated variable MV3 | REAL | 100.0 |
| MV3LoLim | Low limit of manipulated variable MV3 | REAL | 0.0 |
| MV3Man* | Manual value: Operator input for setting the manipulated variable MV3 in manual mode | REAL | 0.0 |
| MV3ManHiLim | High limit of manipulated variable MV3 in manual mode | REAL | 100.0 |
| MV3ManLoLim | Low limit of manipulated variable MV3 in manual mode | REAL | 0.0 |
| MV3RaLim | Gradient limit of the manipulated variable MV3 per sampling step | REAL | 100.0 |
| MV3Traj | Trajectory value that is added to the manipulated variable MV3 | REAL | 0.0 |
| MV3Trk | Tracking value for the manipulated variable MV3 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV3TrkOn | 1 = Tracking of manipulated variable MV3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV4_Unit | Unit of measure for manipulated variable 4 | INT | 1342 |
| MV4Excite | MV4 for automatic process trigger | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV4HiLim | High limit of manipulated variable MV4 | REAL | 100.0 |
| MV4LoLim | Low limit of manipulated variable MV4 | REAL | 0.0 |
| MV4Man* | Manual value: Operator input for setting the manipulated variable MV4 in manual mode | REAL | 0.0 |
| MV4ManHiLim | High limit of manipulated variable MV4 in manual mode | REAL | 100.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MV4ManLoLim | Low limit of manipulated variable MV4 in manual mode | REAL | 0.0 |
| MV4RaLim | Gradient limit of the manipulated variable MV4 per sampling step | REAL | 100.0 |
| MV4Traj | Trajectory value that is added to the manipulated variable MV4 | REAL | 0.0 |
| MV4Trk | Tracking value for the manipulated variable MV4 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV4TrkOn | 1 = Tracking of manipulated variable MV4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = Allocated by SIMATIC BATCH | BOOL | 0 |
| OosLi | 1 = Out of service, via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = Out of service, via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations | DWORD | 16#00000000 |
| OptimOffOp | 1 = Disable optimization, normal setpoints SP1...SP4 in effect | BOOL | 0 |
| OptimOnOp | 1 = Enable optimization, optimized setpoints SP1Out ... SP4Out in effect | BOOL | 0 |
| OS_Perm | I/O for operator permissions | STRUCT<br>• Bit 0: BOOL<br>• Bit 25: BOOL<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| PredictMode | 1 = "Prediction mode" enabled, prediction only, no intervention in the process | BOOL | 0 |
| PreFilt1 | Settling time [s] of the setpoint filter for setpoint SP1 | REAL | 0.0 |
| PreFilt2 | Time constant [s] of the setpoint filter for setpoint SP2 | REAL | 0.0 |
| PreFilt3 | Time constant [s] of the setpoint filter for setpoint SP3 | REAL | 0.0 |
| PreFilt4 | Time constant [s] of the setpoint filter for setpoint SP4 | REAL | 0.0 |
| Restart* | 1 = Restart of the block and adoption of the data from the user block that is entered at the input parameter DB_No | BOOL | 1 |
| SafePos1 | Neutral position for MV1 | BOOL | 0 |
| SafePos2 | Neutral position for MV2 | BOOL | 0 |
| SafePos3 | Neutral position for MV3 | BOOL | 0 |
| SafePos4 | Neutral position for MV4 | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 1.0 |
| SelFp1 | 1 = Call a block saved in this parameter as an additional faceplate in the standard view | ANY | - |
| SelFp2 | 1 = Call a block saved in this parameter as an additional faceplate in the preview | ANY | - |
| SimCV1* | Controlled variable CV1 (process value) which is used with SimOn = 1 | REAL | 0.0 |
| SimCV2* | Controlled variable CV2 (process value) which is used with SimOn = 1 | REAL | 0.0 |
| SimCV3* | Controlled variable CV3 (process value) which is used with SimOn = 1 | REAL | 0.0 |
| SimCV4 | Controlled variable CV4 (process value) which is used with SimOn = 1 | REAL | 0.0 |
| SimCV1Li | Controlled variable CV1 (process value) that is used for SimOnLi.Value = 1 (SimLiOp.Value =1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| SimCV2Li | Controlled variable CV2 (process value) that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| SimCV3Li | Controlled variable CV3 (process value) that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| SimCV4Li | Controlled variable CV4 (process value) that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SP_TrkCV | 1 = Setpoints follow the CVs in manual mode and in tracking | BOOL | 0 |
| SP1* | Setpoint 1<br>SP1.ST=FF: Operable in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| SP1DeadBand | Width of the dead band control of CV1 | REAL | 0.0 |
| SP1HiLim | High limit for setpoint 1 | REAL | 100.0 |
| SP1LoLim | Low limit for setpoint 1 | REAL | 0.0 |
| SP1OptHiLim | High limit for optimization of setpoint 1 | REAL | 100.0 |
| SP1OptLoLim | Low limit for optimization of setpoint 1 | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP2* | Setpoint 2<br>SP2.ST=FF: Operable in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| SP2DeadBand | Width of the dead band control of CV2 | REAL | 0.0 |
| SP2HiLim | High limit for setpoint 2 | REAL | 100.0 |
| SP2LoLim | Low limit for setpoint 2 | REAL | 0.0 |
| SP2OptHiLim | High limit for optimization of setpoint 2 | REAL | 100.0 |
| SP2OptLoLim | Low limit for optimization of setpoint 2 | REAL | 0.0 |
| SP3* | Setpoint 3<br>SP3.ST=FF: Operable in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| SP3DeadBand | Width of the dead band control of CV3 | REAL | 0.0 |
| SP3HiLim | High limit for setpoint 3 | REAL | 100.0 |
| SP3LoLim | Low limit for setpoint 3 | REAL | 0.0 |
| SP3OptHiLim | High limit for optimization of setpoint 3 | REAL | 100.0 |
| SP3OptLoLim | Low limit for optimization of setpoint 3 | REAL | 0.0 |
| SP4* | Setpoint 4<br>SP4.ST=FF: Operable in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| SP4DeadBand | Width of the dead band control of CV4 | REAL | 0.0 |
| SP4HiLim | High limit for setpoint 4 | REAL | 100.0 |
| SP4LoLim | Low limit for setpoint 4 | REAL | 0.0 |
| SP4OptHiLim | High limit for optimization of setpoint 4 | REAL | 100.0 |
| SP4OptLoLim | Low limit for optimization of setpoint 4 | REAL | 0.0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutAct | 1 = Automatic mode is active<br>0 = Manual mode is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CV1Out | Output of manipulated variable 1 (process value) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| CV2Out | Output of manipulated variable 2 (process value) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| CV3Out | Output of manipulated variable 3 (process value) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| CV4Out | Output of manipulated variable 4 (process value) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. You can find information via the error numbers that are output by this block at ModPreCon error handling (Page 989) | INT | -1 |
| ErrorOpt | Error number of the integrated optimization function, see the description of function block LPOptim | INT | 0 |
| Fut1_y1... Fut1_y5 | Prediction of free movement of CV1 for five future points in time within the prediction horizon | REAL | 0 |
| Fut2_y1... Fut2_y5 | Prediction of free movement of CV2 for five future points in time within the prediction horizon | REAL | 0 |
| Fut3_y1... Fut3_y5 | Prediction of free movement of CV3 for five future points in time within the prediction horizon | REAL | 0 |
| Fut4_y1... Fut4_y5 | Prediction of free movement of CV4 for five future points in time within the prediction horizon | REAL | 0 |
| J_Actual | Current value of the performance criterion | REAL | 0 |
| Loop1Closed | 1 = Control loop for CV1 closed<br>0 = Control loop for CV1 open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Loop2Closed | 1 = Control loop for CV2 closed<br>0 = Control loop for CV2 open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Loop3Closed | 1 = Control loop for CV3 closed<br>0 = Control loop for CV3 open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Loop4Closed | 1 = Control loop for CV4 closed<br>0 = Control loop for CV4 open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ManAct | 1 = Manual mode active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Release | Release for maintenance:<br>1 = Release by OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV1 | Control variable 1 (control signal) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV1HiAct | 1 = High limit of manipulated variable 1 reached or exceeded | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV1LoAct | 1 = Low limit of manipulated variable 1 reached or exceeded | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV1Pred | One-step prediction for MV1 in the "Prediction without control response" mode | REAL | 0 |
| MV2 | Manipulated variable 2 (control signal) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV2HiAct | 1 = High limit of manipulated variable 2 reached or exceeded | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV2LoAct | 1 = Low limit of manipulated variable 2 reached or exceeded | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV2Pred | One-step prediction for MV2 in the "Prediction without control response" mode | REAL | 0 |
| MV3 | Manipulated variable 3 (control signal) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV3HiAct | 1 = High limit of manipulated variable 3 reached or exceeded | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV3LoAct | 1 = Low limit of manipulated variable 3 reached or exceeded | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV3Pred | One-step prediction for MV3 in the "Prediction without control response" mode | REAL | 0 |
| MV4 | Manipulated variable 4 (control signal) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MV4HiAct | 1 = High limit of manipulated variable 4 reached or exceeded | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV4LoAct | 1 = Low limit of manipulated variable 4 reached or exceeded | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV4Pred | One-step prediction for MV4 in the "Prediction without control response" mode | REAL | 0 |
| NumberCVs | Number of controlled variables (process values) used | INT | 0 |
| NumberDVs | Number of disturbances used | INT | 0 |
| NumberMVs | Number of manipulated variables used | INT | 4 |
| OosAct | 1 = Block is "out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OptimAct | 1 = Optimization is active | BOOL | 0 |
| OptimAvailable | 1 = Optimization available, 0 = Optimization not available, because old user data block is loaded | BOOL | 0 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| PrediHorizon | Prediction horizon [s] | REAL | 0 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP1OpOut | Copy of the operable setpoint 1 for step-enabling | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP2OpOut | Copy of the operable setpoint 2 for step-enabling | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP3OpOut | Copy of the operable setpoint 3 for step-enabling | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP4OpOut | Copy of the operable setpoint 4 for step-enabling | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP1Out | Setpoint 1 used by controller | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP2Out | Setpoint 2 used by controller | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP3Out | Setpoint 3 used by controller | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP4Out | Setpoint 4 used by controller | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 | DWORD | 16#00 |
| Status2 | Status word 2 | DWORD | 16#00 |

### See also

Description of ModPreCon (Page 970)
ModPreCon messaging (Page 990)
ModPreCon block diagram (Page 1001)
ModPreCon modes (Page 975)
Neutral position for motors, valves and controllers (Page 54)
Opening additional faceplates (Page 234)
Description of OpStations (Page 457)

## 6.6.7 ModPreCon block diagram

### ModPreCon block diagram

A block diagram is not provided for this block.

### See also

ModPreCon I/Os (Page 990)
ModPreCon messaging (Page 990)
ModPreCon error handling (Page 989)
ModPreCon functions (Page 976)

ModPreCon modes (Page 975)

Description of ModPreCon (Page 970)

## 6.6.8 Operator control and monitoring

### 6.6.8.1 ModPreCon views

#### Views of the ModPreCon block

The block ModPreCon provides the following views:

- ModPreCon standard view (Page 1003)
- ModPreCon trend view (Page 1013)
- ModPreCon parameter view (Page 1008)
- ModPreCon parameter view channel 1 to 4 (Page 1010)
- ModPreCon preview (Page 1011)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for ModPreCon (Page 1014)

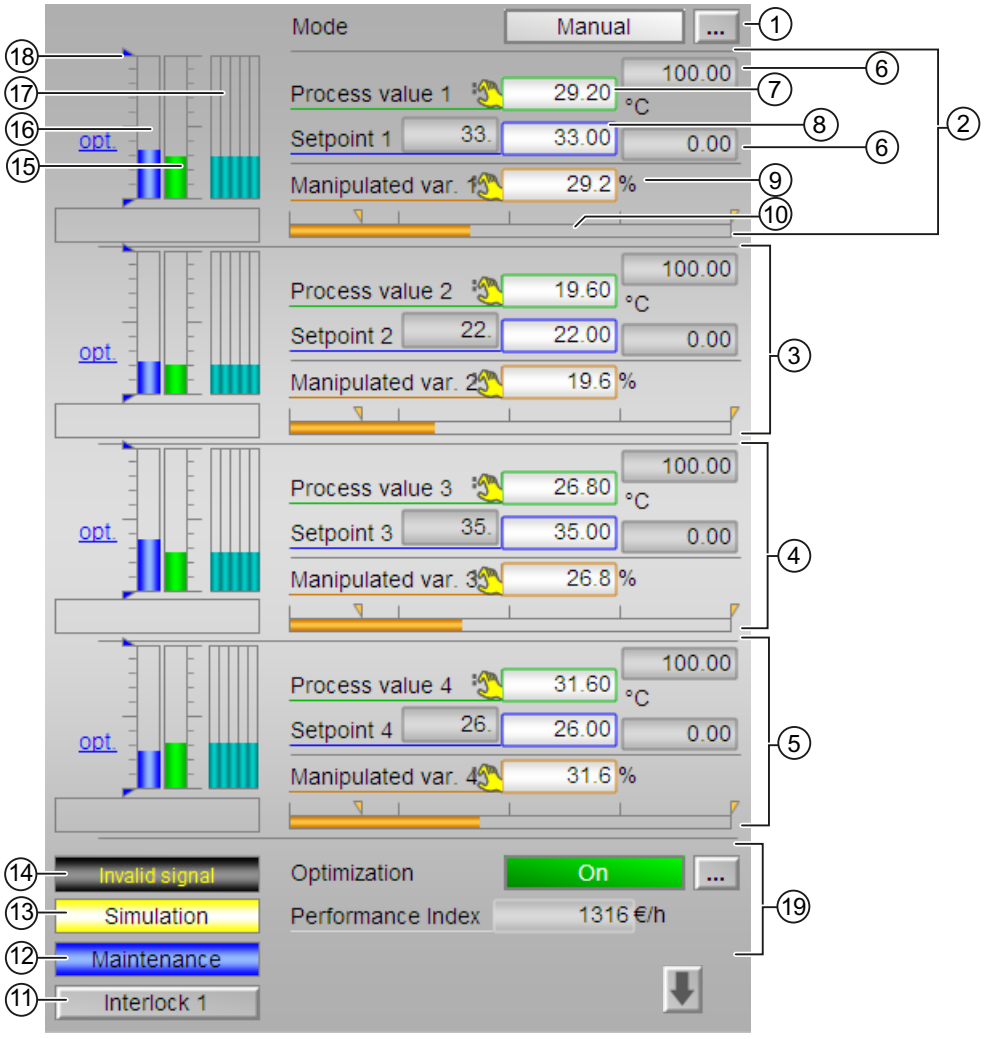
Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

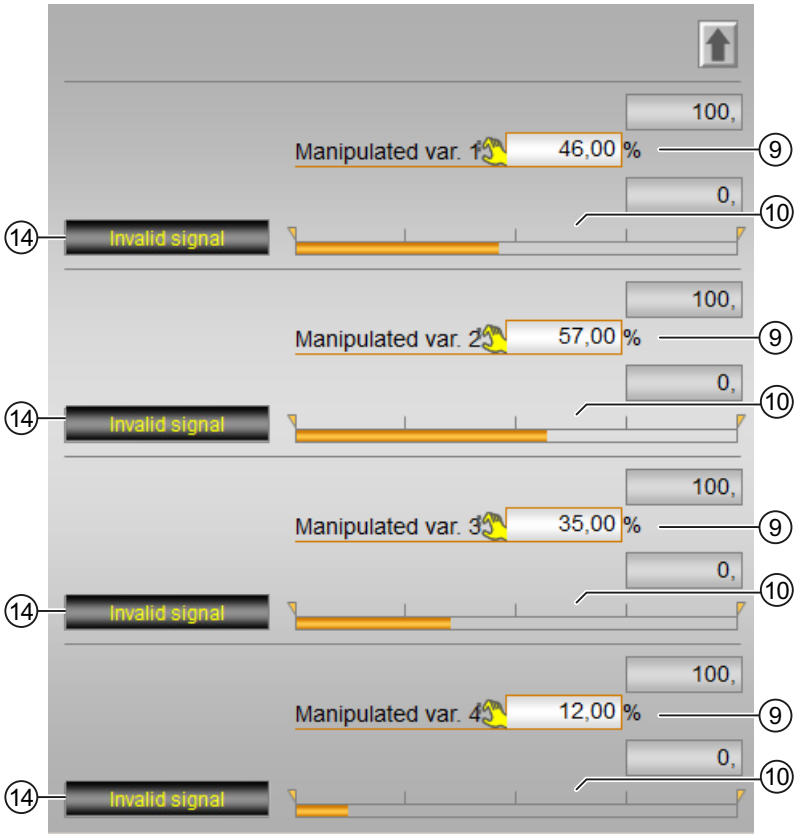
#### See also

Trend view (Page 341)

### 6.6.8.2 ModPreCon standard view

## ModPreCon standard view






The standard view has an upper half and a lower half. You can change between the two halves with the arrow keys. The upper half shows all available controlled variable channels with their setpoints, while the lower half shows all available manipulated variable channels.

**Upper screen half (controlled variables)**

## (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 79)
- Automatic mode (Page 79)
- Out of service (Page 77)

Refer to section Switching operating states and operating modes (Page 289) for information on switching the operating mode.

## (2), (3), (4) and (5) Displaying and switching for values for channels 1 to 4

This area always has the same layout for channels 1 to 4:

**(6) High and low scale range for the process value**

These values provide information on the display range for the bar graph of the process value. The scale range is defined in the engineering system.

**(7) Displaying and changing the process value including signal status**

This area shows the current process value with the corresponding signal status.

In the object properties (I/Os PVx_Out > Identifier) of the block in CFC, you can specify the text to be displayed for these parameters.

**(8) Displaying and changing the setpoint including signal status**

This area shows the current setpoint with the corresponding signal status. Refer to the Changing values (Page 291) section for information on changing the setpoint.

### (11) Navigation button for switching to the standard view of any faceplate

Use this navigation button to open the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234).

### (12) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this is available in section Release for maintenance (Page 70) Display area for block states.

### (13) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"
- "Process excitation"
  The automatic process trigger is fed forward using the upstream block AutoExcitation for recording learning data for the MPC configurator. The manipulated variable step changes are added to the manipulated values 1 to 4 according to schedule. Avoid external disturbances to the process while the process trigger is running. The manipulated variables can be changed manually while the process trigger is running.

You can find additional information on this in the section Simulating signals (Page 64).

### (14) Display area for block states

- "Internal error"
- "Invalid Signal"

### (15) Display area for block states

- "Tracking on"

### (16) Bar graph for the process value 1

There is a bar graph for the process value for every channel 1 to 4.

This area shows the current process value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (17) Bar graph for the setpoint 1

There is a bar graph for the setpoint for every channel 1 to 4.

This area shows the current setpoint in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (18) Prediction of free movement

This area shows you the prediction of free movement in the form of a bar graph. For each channel from 1 to 4, there is a bar graph for the prediction of free movement, that is, for the future behavior of the process within the overall prediction horizon, under the assumption that all manipulated variables are frozen at their current values.

This is why the prediction of free movement is only displayed in manual mode or in automatic mode with active tracking function.

The value range of the bar graph matches the value range of the assigned setpoint and current value bar.

ModPreCon functions (Page 976)

### (19) Displaying the limits

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the engineering system (ES).

### (20) Static operating point optimization

Activate the optimization using the button at bottom right. Activation means that the optimized setpoints SP1Out...SP4Out are actually used instead of the SP1...SP4 setpoints specified in the faceplate for the closed-loop control. (The actual calculation of the optimum setpoints depends on this, and is only performed if one of the input variables for the optimization has changed.) The current value the economic performance criterion J appears in the display field below.

When optimization is enabled, the optimum setpoints are displayed on the setpoint bar as small, horizontal lines and highlighted with the abbreviation "opt.". The numerical values of the optimum setpoints are then displayed left of the input fields for the setpoints.

**Lower screen half (manipulated variables)**

### (9) Displaying and changing the manipulated variable including signal status

This area shows the current manipulated variable with the corresponding signal status. Refer to the Changing values (Page 291) section for information on changing the manipulated variable. You can only make a change in manual mode.

In the object properties (I/Os MVx > Identifier) of the block in CFC, you can specify the text to be displayed for these parameters.

### (10) Bar graph for the manipulated variable with limit display

This area shows the current manipulated variable in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES):

- Limits: `MVxHiLim` and `MVxLoLim`
- Display area: `MVxManHiLim` and `MVxManLoLim`

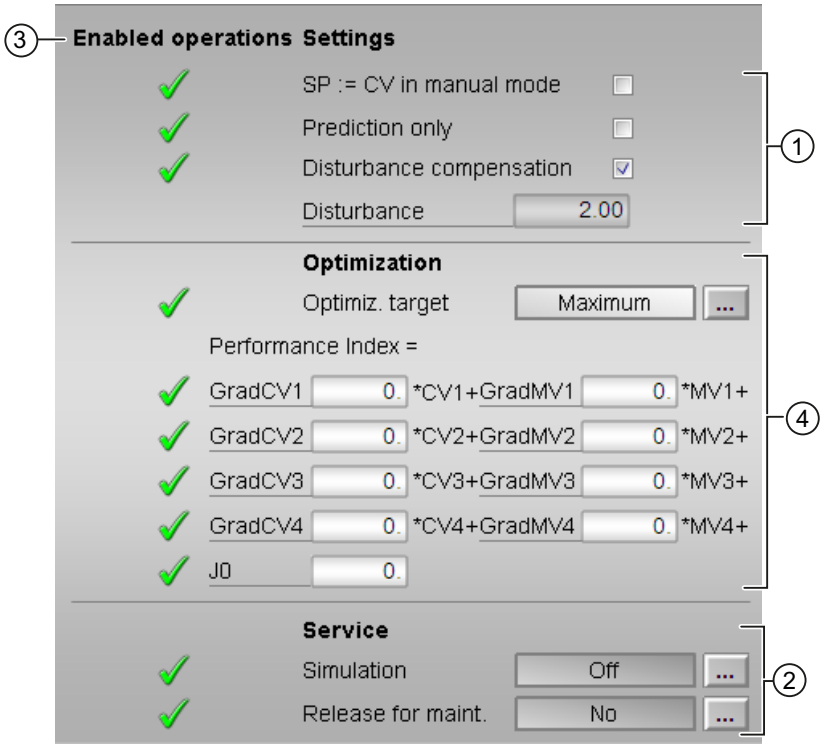
### (14) Display for block states

There is a display for the states of the block for every channel 1 to 4:

- "Tracking"

### 6.6.8.3 ModPreCon parameter view

#### Parameter view of ModPreCon




#### (1) Settings

You can activate the following functions for the controller in this area:

- "SP := CV in manual mode": ☑ Bumpless switchover from "manual mode" to "automatic mode"
- "Prediction only" activate this special "operating mode" by selecting the check box. The controller then only listens in on the process and indicates what it would like to do in the next sampling step without actively intervening in the process
- "Disturbance compensation": ☑ Select disturbance feedforward
- "Disturbance variable"

You cannot change the disturbance variable, it can only be displayed.

In the object properties (I/Os DV1 > Identifier) of the block in CFC, you can specify the text to be displayed for this parameter.

#### (2) Service

You can select the following functions in this area:

- "Simulation"
- "Release for maintenance"

Refer to the Switching operating states and operating modes (Page 289) section for more on this.

You can find information on this area in the section:

- Simulating signals (Page 64)
- Release for maintenance (Page 70)

### (3) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

### (4) Optimization

**Direction of the optimization (minimize or maximize)**

By default, the optimizer seeks to maximize the performance function, in the assumption that it is dealing with economic yield. If you want to search a minimum, however, because you are dealing with costs or consumption values, click this button.

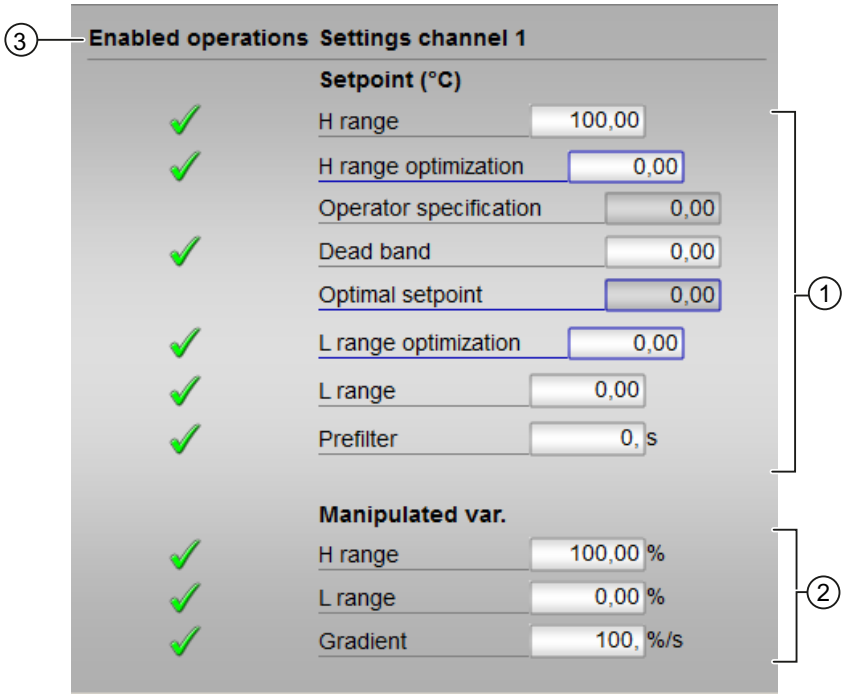
**Specification of performance criterion for the operating point optimization**

The performance criterion consists of a weighted sum of all manipulated and controlled variables. For each manipulated variable and controlled variable, enter the appropriate weighting factor, i.e. the coefficient of the gradient vector. Zero means that the value of the corresponding manipulated variable or controlled variable no direct influence on the economic yield. If the controller has less than four manipulated variables or controlled variables, the irrelevant variables are hidden automatically.

### 6.6.8.4 ModPreCon parameter view channel 1 to 4

#### Parameter view channel 1 to 4 for ModPreCon

The layout of the parameter view for channels 1 to 4 is always identical:




#### (1) Displaying and changing the limit parameters for the setpoint

You can change the following parameters for the setpoint in this area:

- "H range": High limit for setpoint operation
- "H range optimization": High limit for optimizing the setpoint
- "Operator input": Display of the setpoint entered in the standard view, cannot be operated here.
- "Dead band": Dead band (Page 67), Error signal generation and dead band section
- "Optimal setpoint": Calculated by the optimization, cannot be operated
- "L range optimization": Low limit for optimizing the setpoint
- "L range": Low limit for setpoint operation
- "Prefilter": ModPreCon functions (Page 976), Setpoint filter section

You can find additional information on this in the section Changing values (Page 291) .

### (2) Displaying and changing the limit parameters for the manipulated variable

You can change the following parameters for the manipulated variable in this area:

- "H range": Upper limit of the manipulated variable for automatic mode
- "L range": Low limit of manipulated variable for automatic mode
- "Gradient limit": Maximum (absolute) change in the manipulated variable per sampling step

### (3) Enabled operation

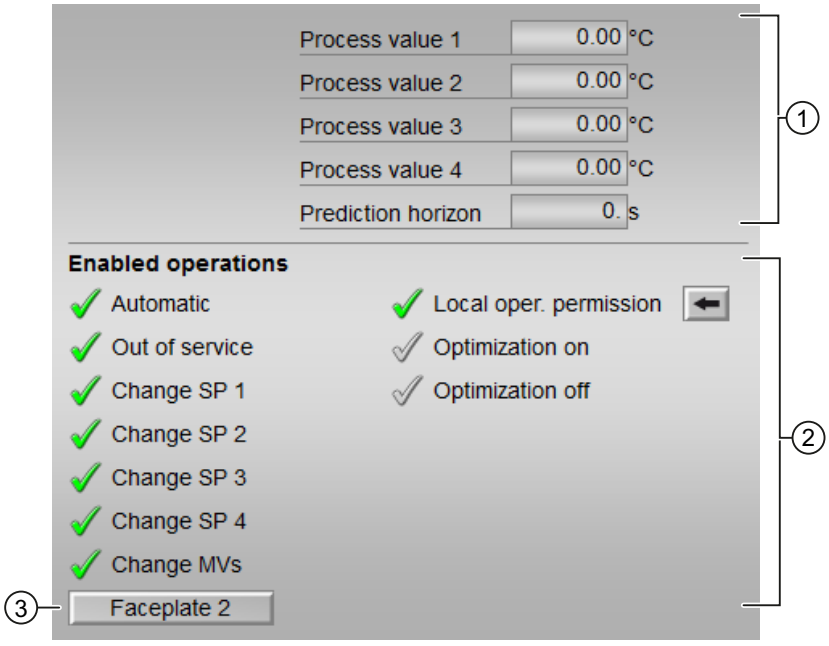
This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

## 6.6.8.5 ModPreCon preview

### Preview for ModPreCon




### (1) Process value

This area displays the real process values (`PVx`) and the prediction horizon.

In the object properties (I/Os PVxOut > Identifier) of the block in CFC, you can specify the text to be displayed for these parameters.

**Prediction horizon**

The prediction horizon specifies how far the controller looks into the furutre of its calculation.

### (2) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

The following enabled operations for parameters are shown here:

- "Automatic": You can switch to "automatic mode".
- "Out of service": You can switch to "Out of service" operating mode.
- " "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. You can find additional information on this in the section Operator control permissions (Page 285) .
- "Change SP1": You can change the setpoint 1
- "Change SP2": You can change the setpoint 2
- "Change SP3": You can change the setpoint 3
- "Change SP4": You can change the setpoint 4
- "Change MVs": You can change the manipulated variables

**Note**

The OS operator must always be able to switch to "manual mode". That is why the switch to "manual mode" is not shown here in the faceplate.

### (3) Navigation button for switching to the standard view of any faceplate

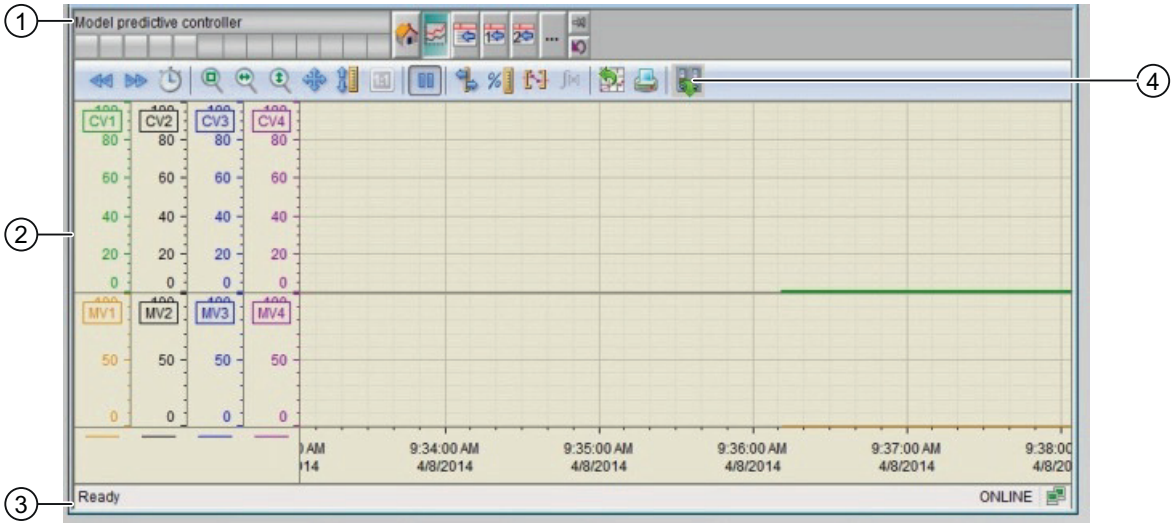
Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234) .

### 6.6.8.6 ModPreCon trend view

## ModPreCon trend view

There is a block-specific trend view for ModPreCon which is supplied as file @PG_APL_TrendMPC.pdl and which you can modify if necessary.




**(1)** Toolbar

**(2)** Display area for trends

**(3)** Status bar

**(4)** Button for switching between archive tags and online tags. The status bar shows if the trend view is working with online data or archive data.

The Export button is only visible and operable with the "Higher-level process control" operating permission.

For additional information about the trend view, refer to the *WinCC Information System* Online Help.

The trend view is divided into two screen halves.

The upper screen half shows all controlled variables with their associated setpoints. The setpoint is shown in the same color as the associated process value to allow the assignment to be identified straight away. Setpoints are dashed lines, process values are bold lines. If a controlled variable is exactly on the setpoint, it hides the setpoint.

The lower screen half shows all manipulated variables.

Both screen halves use the same color sequence for the individual channels. The sequence starts at channel 1 with green (standard color for the process value with the PID controller) and then goes through the color spectrum from top to bottom as far as gray and black. Each channel has its own y-axis in the corresponding color.

### See also

ModPreCon views (Page 1002)

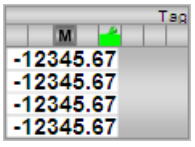
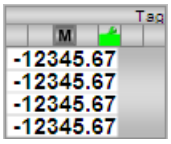
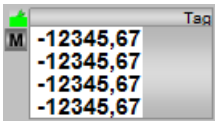
## 6.6.8.7 Block icon for ModPreCon

### Block icons for ModPreCon

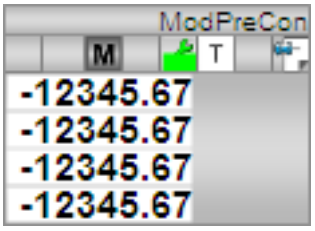
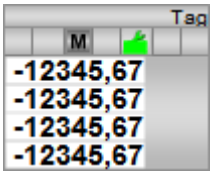
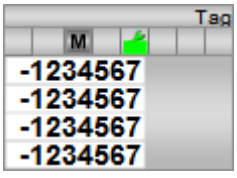
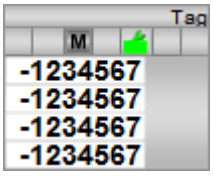
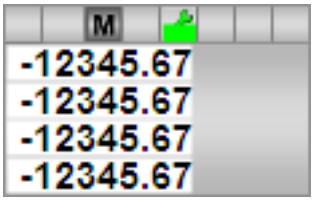
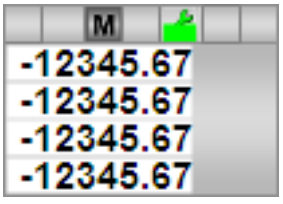
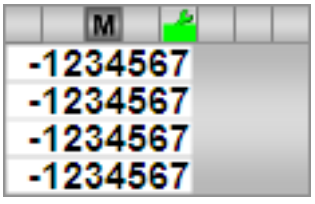
A variety of block icons are available with the following functions:

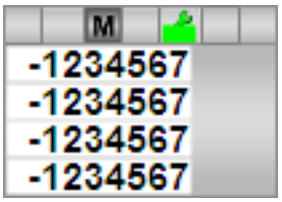
- Process tag type
- Operating modes
- Signal status, release for maintenance
- Tracking
- Memo display
- Process value (black, with and without decimal places)

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | Block icon in the full display |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | |
|  | 6 | |
|  | 7 | |

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 8 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269).

# 6.7 PIDConL - Continuous PID controller (Large)

## 6.7.1 Description of PIDConL

### Object name (type + number) and family

Type and number: FB 1874

Family: Control

### Area of application for PIDConL

The block is used for the following applications:

- Fixed setpoint control
- Cascade control
- Ratio control
- Split-range control
- Smith predictor closed-loop control
- Override control (override)

### How it works

The block is a PID controller with continuous output signal (manipulated variable). It is used to activate a final controlling element with continuous action input.

The block functions following the PID algorithm with a delayed D component and an integrator with double precision.

The block is suitable for controlling sluggish control loops, for example, for temperatures and filling levels, and high-speed control loops, for example, for flow rates and speed. For a given CPU, a compromise has to be made between the number of controllers and the frequency with which the individual controllers have to be processed. The faster the modulated control loops are, i.e. the more frequently the manipulated variables have to be calculated per time unit, the lower the number of controllers that can be installed.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

For the control loop monitoring to work as planned in the trend view of the controller faceplates, the

```
S7_xarchive:='Value, shortterm;'
```

attributes in the process tag types for control loops at the controller function block must be set for the following tags:

- Input parameters:
  - CPI_In
- Output parameters
  - MV
  - MV_HiAct
  - MV_LoAct
  - LoopClosed
  - SP
  - PV_Out
  - PV_ToleHi
  - PV_ToleLo

For the PIDConL block, the Advanced Process Library contains templates for process tag types as examples and there is a example project (APL_Example_xx, xx designates the language variant) containing different application cases for this block. Several application cases are simulated in the example project and serve to explain how the block works.

Examples of process tag types:

- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 2822)
- Split-range controller with control loop monitoring through ConPerMon (SplitrangeControl) (Page 2833)
- Ratio control with control loop monitoring through ConPerMon (RatioControl) (Page 2836)
- Cascade control with control loop monitoring through ConPerMon (CascadeControl) (Page 2840)
- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 2825)
- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 2827)
- Override control (Page 2846)
- PID controller with Smith predictor (SmithPredictorControl) (Page 2830)
- PID controller for PA/FF devices (PIDControl_Lean_Fb) (Page 2821)

Application cases in example project:

- Process simulation including noise generator (ProcSimC; ProcSimS) (Page 2871)
- Cascade control of temperature by using the heat flow (CascadeSim) (Page 2873)
- Control loop monitoring for simulation with colored noise (ConPerMonSim) (Page 2875)
- Feedforward control to compensate a measurable disturbance variable (DisturbCompSim) (Page 2875)
- Operating point-oriented adaptation of parameters (gain scheduling) for non-linear processes (GainSchedSim) (Page 2876)

- Override control on a pipeline (OverrideSim) (Page 2877)
- Smith predictor for a dead time system (SmithPredictorSim) (Page 2877)
- Filtering of noisy measured values in a control loop (SigSmoothSim) (Page 2877)

### Startup characteristics

Use the `Feature` bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

The following output parameters are written with the corresponding input parameters:

- `PV_HysOut`
- `PV_AH_Out`
- `PV_WH_Out`
- `PV_TH_Out`
- `PV_AL_Out`
- `PV_WL_Out`
- `PV_TL_Out`

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section PIDConL I/Os (Page 1039).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `AutAct.Value` |
| 6 | Invalid signal status |
| 7 | `ManAct.Value` |
| 8 | `SP_ExtAct.Value` |
| 9 | `MV_ForOn.Value` |
| 10 | `MV_TrkOn.Value` AND NOT (`ManAct.Value` OR `OosAct.Value` OR `MV_ForOn.Value`) |
| 11 | `MV.Value` > `ManLoLim` |
| 12 | `SimLiOp.Value` |
| 13 | "Interlock" button is enabled OR `Intlock.ST` ≠ 16#FF |
| 14 | 0 = Open padlock in the block icon<br>1 = Closed padlock in block icon |
| 15 - 17 | Not used (can be set to 1 if `Intl.Status` = 16#60) |
| 18 | `SimOn` AND `ManAct` |

| Status bit | Parameter |
|---|---|
| 19 | AdvCoAct |
| 20 | 1 = Input parameter Rbk is not interconnected (RbkOut.ST = 16#FF) |
| 21 | NegGain |
| 22 | 1 = SP ramp active |
| 23 | OptimEn |
| 24 | OptimOcc |
| 25 | Not used (can be set to 1 if Intl.Status = 16#60) |
| 26 | Display of BypassAct.Value in faceplate (display and operator controls) and block icon |
| 27 | AdvCoModSP |
| 28 | Not used (can be set to 1 if Intl.Status = 16#60) |
| 29 | BypProt |
| 30 | Dead band is temporarily disabled |
| 31 | Bypass information for display in block icon |

### Status word allocation for Status2 parameter

| Status bit | Parameter |
|---|---|
| 0 | MsgLock |
| 1 | PV_AH_Act.Value |
| 2 | PV_WH_Act.Value |
| 3 | PV_TH_Act.Value |
| 4 | PV_TL_Act.Value |
| 5 | PV_WL_Act.Value |
| 6 | PV_AL_Act.Value |
| 7 | PV_AH_En OR Feature.Bit31 |
| 8 | PV_WH_En OR Feature.Bit31 |
| 9 | PV_TH_En OR Feature.Bit31 |
| 10 | PV_TL_En OR Feature.Bit31 |
| 11 | PV_WL_En OR Feature.Bit31 |
| 12 | PV_AL_En OR Feature.Bit31 |
| 13 | PV_AH_MsgEn |
| 14 | PV_WH_MsgEn |
| 15 | PV_TH_MsgEn |
| 16 | PV_TL_MsgEn |
| 17 | PV_WL_MsgEn |
| 18 | PV_AL_MsgEn |
| 19 | ER_AH_Act.Value |
| 20 | ER_AL_Act.Value |
| 21 | ER_AH_En OR Feature.Bit31 |
| 22 | ER_AL_En OR Feature.Bit31 |
| 23 | ER_AH_MsgEn |

| Status bit | Parameter |
|---|---|
| 24 | ER_AL_MsgEn |
| 25 | RbkWH_Act.Value |
| 26 | RbkWL_Act.Value |
| 27 | RbkWH_En OR Feature.Bit31 |
| 28 | RbkWL_En OR Feature.Bit31 |
| 29 | RbkWH_MsgEn |
| 30 | RbkWL_MsgEn |
| 31 | MS_RelOp |

### Status word allocation for Status3 parameter

| Status bit | Parameter |
|---|---|
| 0 | Effective signal 1 of the message block connected via EventTsIn |
| 1 | Effective signal 2 of the message block connected via EventTsIn |
| 2 | Effective signal 3 of the message block connected via EventTsIn |
| 3 | Effective signal 4 of the message block connected via EventTsIn |
| 4 | Effective signal 5 of the message block connected via EventTsIn |
| 5 | Effective signal 6 of the message block connected via EventTsIn |
| 6 | Effective signal 7 of the message block connected via EventTsIn |
| 7 | Effective signal 8 of the message block connected via EventTsIn |
| 8 | Delay of the PV_AH_Lim message |
| 9 | Delay of the PV_WH_Lim message |
| 10 | Delay of the PV_TH_Lim message |
| 11 | Delay of the PV_TL_Lim message |
| 12 | Delay of the PV_WL_Lim message |
| 13 | Delay of the PV_AL_Lim message |
| 14 | Delay of the ER_AH_Lim message |
| 15 | Delay of the ER_AL_Lim message |
| 16 | Collection of message delays |
| 17 | BypassAct.Value |
| 18 - 19 | Not used |
| 20 | 1= Suppressed messages |
| 21 - 23 | Not used |
| 24 | Feature2.Bit2: Separate bypass signal |
| 25 | Hidden bypass signal in Intlock |
| 26 | LockAct.Value |
| 27 | SP_UpRaAct, SP_DnRaAct limits enabled for gradient mode (SP_RateOn = 1) |
| 28 | GrpErr.Value |
| 29 | RdyToStart.Value |
| 30 | 1 = Input parameter MV_ChnST is interconnected |
| 31 | Internal error |

### Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 - 7 | Effective signal 9..16 of the message block connected via `EventTsIn` |
| 8 - 28 | Not used |
| 29 | 1 = Input parameter `SP_LiRmpTime` is not interconnected (`SP_LiRmpTime.ST = 16#FF`) |
| 30 | Auxiliary value 1 visible |
| 31 | Auxiliary value 2 visible |

### See also

PIDConL functions (Page 1024)

PIDConL messaging (Page 1036)

PIDConL block diagram (Page 1058)

PIDConL error handling (Page 1034)

PIDConL modes (Page 1022)

## 6.7.2 PIDConL modes

### PIDConL operating modes

The block can be operated using the following modes:

- Automatic mode (Page 79)
- Manual mode (Page 79)
- Program mode for controllers (Page 85)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and bumpless switchover in the Manual and automatic mode for control blocks (Page 79) section.

### "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the Manual and automatic mode for control blocks (Page 79) section.

### "Program mode for controllers"

General information on "Program mode for controllers" is available in the section Program mode for controllers (Page 85).

### "Out of service"

You can find general information about the "Out of service" mode in the Out of service (Page 77) section.

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit5` =0 and `Feature.Bit5` =1 is as shown below:

| Output Parameters | **`Feature.Bit5` =0** | **`Feature.Bit5` =1** |
|---|---|---|
| RbkOut.ST | 16#80 | 16#60 |
| RbkWH_Act.ST | 16#XX | 16#60 |
| RbkWL_Act.ST | 16#XX | 16#60 |
| SP_ExtAct.ST | 16#80 | 16#60 |
| SP.ST | 16#80 | 16#60 |
| SP_ExtOut.ST | 16#80 | 16#60 |
| PV_Out.ST | 16#80 | 16#60 |
| PV_AH_Act.ST | 16#XX | 16#60 |
| PV_WH_Act.ST | 16#XX | 16#60 |
| PV_TH_Act.ST | 16#XX | 16#60 |
| PV_TL_Act.ST | 16#XX | 16#60 |
| PV_WL_Act.ST | 16#XX | 16#60 |
| PV_AL_Act.ST | 16#XX | 16#60 |
| PV_ToleHi.ST | 16#80 | 16#60 |
| PV_ToleLo.ST | 16#80 | 16#60 |
| ER.ST | 16#80 | 16#60 |
| ER_AH_Act.ST | 16#XX | 16#60 |
| ER_AL_Act.ST | 16#XX | 16#60 |

**Note**

The status of `RbkOut` will be 16#FF without Rbk. when `Feature.Bit5` = 0.

### See also

PIDConL block diagram (Page 1058)

PIDConL I/Os (Page 1039)

PIDConL messaging (Page 1036)

PIDConL error handling (Page 1034)

PIDConL functions (Page 1024)

Description of PIDConL (Page 1017)

## 6.7.3 PIDConL functions

### Functions of PIDConL

The functions for this block are listed below.

### Generation of manipulated variables

The manipulated variable `MV` can be generated as follows:

| MV_For-On | Int-lock | ManAct | MV_Trk On | AdvCoAct AND NOT AdvCoModSP | MV = | Limit monitoring | State |
|---|---|---|---|---|---|---|---|
| 1 | - | - | - | - | `MV_Forced` | none | Forced tracking through constraint without limitation |
| 0 | 1 | - | - | - | Neutral position: depends on `SafePos`, `SafePos2` | none | Interlock state |
| 0 | 0 | 1 | - | - | `Man` | `ManHiLim` `ManLoLim` | Manual mode, set by the operator |
| 0 | 0 | 0 | 1 | - | `MV_Trk` | `MV_HiLim` `MV_LoLim` | Tracking with limitation |
| 0 | 0 | 0 | 0 | 1 | `AdvCoMV` | `MV_HiLim` `MV_LoLim` | Higher-level program mode |
| 0 | 0 | 0 | 0 | 0 | `P_Part` + `I_Part` + `D_Part` + `FFwd` | `MV_HiLim` `MV_LoLim` | Automatic mode (PID algorithm) |

If the controller is in "out of service" mode, the output parameter `MV` is set to the last valid value in manual mode or the neutral position manipulated variable depending on the `Feature` Bit (Neutral position manipulated variable takes effect at startup (Page 176)). Refer to the Out of service (Page 77) section for more on this.

### Tracking and limiting a manipulated variable

The block provides the standard function Tracking and limiting a manipulated variable (Page 222).

Tracking has a higher priority than interlock in this block.

### Neutral position

The block provides the standard function Neutral position for motors, valves and controllers (Page 54).

### Group error

This block provides the standard function Outputting group errors (Page 130).

The following parameters are taken into consideration when forming the group error:

- `CSF`

### Outputting a signal for start readiness

This block provides the standard function Outputting a signal for start readiness (Page 59).

### "Actuator active" information

If the manipulated variable `MV` is greater than the minimum manual limit `ManLoLim`, this is recognized as actuator active. This status can be used to indicate a customized symbol in the process image, for example, and is saved in the status word (see Status word section in Description of PIDConL (Page 1017)).

### Limit monitoring of position feedback

The block provides the standard function Limit monitoring of the feedback (Page 101).

### Group display SumMsgAct for limit monitoring, CSF and ExtMsgx

The block provides the standard function Group display for limit monitoring, CSF and ExtMsgx (Page 92).

### External/internal setpoint specification

The block provides the standard function Setpoint specification - internal/external (Page 136).

### Setpoint limiting for external setpoints

The block provides the standard function Setpoint limiting for external setpoints (Page 222).

### Gradient limit of the setpoint

The block provides the standard function Gradient limit of the setpoint (Page 132).

### Using setpoint ramp

The block provides the standard function Using setpoint ramp (Page 131).

### Tracking setpoint in manual mode

The block provides the standard function Tracking setpoint in manual mode (Page 222).

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Process value (`SimPV, SimPV_Li`)
- Position feedback (`SimRbk, SimRbkLi`)

### Bypass function

This block provides the standard function Bypassing signals (Page 115).

### Limit monitoring of the process value

The block provides the standard function Limit monitoring of the process value (Page 93) with the alarm delay type Two time values per limit pair (Page 227). With the `Feature` Bit Separate delay times for each alarm (Page 181), the alarm delay type Two time values for each individual limit (Page 228) can be activated.

### Providing limits at the output

For further connections to the other blocks, the following input parameters are also displayed with the corresponding output parameters:

- `PV_HysOut` := `PV_Hyst`
- `PV_AH_Out` := `PV_AH_Lim`
- `PV_WH_Out` := `PV_WH_Lim`
- `PV_TH_Out` := `PV_TH_Lim`
- `PV_TL_Out` := `PV_TL_Lim`
- `PV_WL_Out` := `PV_WL_Lim`
- `PV_AL_Out` := `PV_AL_Lim`
- `ER_AH_LimOut` := `ER_AH_Lim`
- `ER_AL_LimOut` := `ER_AL_Lim`

### Error signal generation and dead band

The block provides the standard function Error signal generation and dead band (Page 217).

The `Feature.Bit14` can be used to feedforward an external error signal `ER_Ext`. When the external error signal is activated, `ER_Ext` affects both the dead band and the error signal alarm generation.

### Delay alarm for control deviation at setpoint step changes

The block provides the standard function Delay alarm for control deviation at setpoint step changes (Page 216).

### Limit monitoring of error signal

The block provides the standard function Limit monitoring of setpoint, manipulated variable and control deviation (Page 102). The monitoring of error signal works with the alarm delay type Two time values per limit pair (Page 227). With the `Feature` Bit Separate delay times for each alarm (Page 181), the alarm delay type Two time values for each individual limit (Page 228) can be activated.

### Inverting control direction

The block provides the standard function Inverting control direction (Page 217).

### Physical standardization of setpoint, manipulated variable and process value

Controller gain `Gain` is entered either using a physical variable or as standardized value.

`Gain` as a physical variable:

The standardized variables retain their default values:

- `NormPV.High` = 100 and `NormPV.Low` = 0
- `NormMV.High` = 100 and `NormMV.Low` = 0

The effective gain is:

`GainEff` = `Gain`

Entering a standardized `Gain` (dimensionless):

Change the standardized variables to the actual range of the process values and manipulated variables.

- Internal and external setpoints; the process value and corresponding parameters are entered according to the physical measuring range of the process value.
- The manual value, the tracking value of the manipulated variable, feedforward control and the corresponding parameters are set according to the physical measuring range of the manipulated variable.

The effective gain is:

`GainEff` = (`NormMV.High` - `NormMV.Low`) / (`NormPV.High` - `NormPV.Low`) · `Gain`

### Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 240).

### PID algorithm

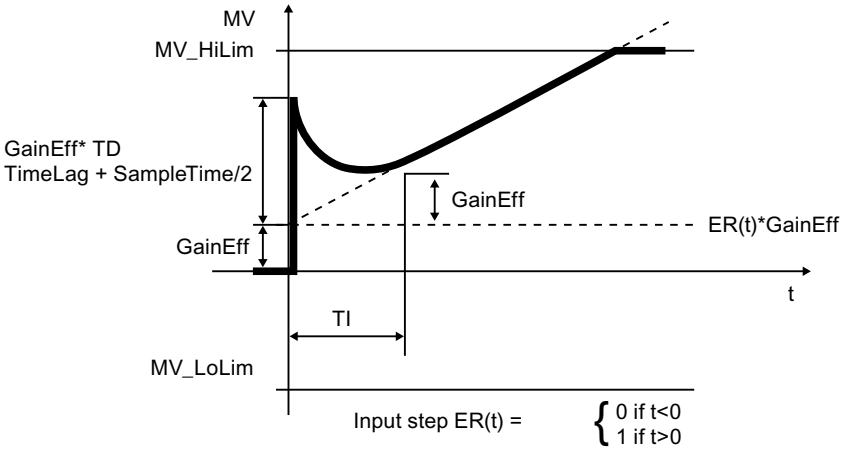
The manipulated variable is generated in automatic mode according to the following algorithm:

`MV` = `GainEff` · (1 + 1 / (`TI` · s) + (`TD` · s) / (1 + `TD` / `DiffGain` · s)) · `ER`

Where:

s = Complex number

The following step response occurs:




**Note**

The formula describes a standard application where P, I and D components are activated and the P and D components are not in the feedback circuit (`PropSel` = `1`, `TI` <> `0` and `IntSel` = `1`, `TD` <> `0` and `DIffSel` = `1`, `DiffToFbk` = `0` and `PropFacSP` = 1).

The D component delay is derived from `TD` / `DiffGain`.

- The P component is displayed at the `P_Part` I/O and can be deactivated using `PropSel` = 0 .
- The I component is displayed at the I_Part I/O and can be deactivated using `TI = 0` or IntSel = 0. In deactivated state, I_Part is specified by MV_Offset and added to the manipulated variable. Make a selection for this value so that the remaining control deviation equals zero at the control loop's typical operating point, at least. `IntSel` is used for temporary deactivation of the I component. The I component is not reactivated until `TI <> 0` and `IntSel = 1`. After the I component is activated, the integrator continues working starting from `MV_Offset`.
- The D component is displayed at the `D_Part` I/O and can be deactivated using `TD` = 0 or `DiffSel` = `0`.

### Structure segmentation at controllers

The block provides the standard function Structure segmentation at controllers (Page 224).

### Anti-windup

The controller has an anti-windup function. The I component is frozen after the manipulated variable has reached limits (`MV_HiLim` or `MV_LoLim`).

### Feedforwarding and limiting disturbance variables

The block provides the standard function Feedforwarding and limiting disturbance variables (Page 223).

### Control zone

The block provides the standard function Using control zones (Page 220).

### Control loop is open (CascaCut)

The CascaCut output indicates whether the control loop within this block is open. In the open control loop, the master controller must be set to tracking mode to avoid until windup behavior. If you use this block in a cascade structure, then connect the output CascaCut to the input MV_TrkOn of the master controller.

CascaCut is active in the following cases:

- Block is not in automatic mode
- MV program mode is active
- Setpoint is not in external mode
- Interlock is active

### Forming the signal status for blocks

The block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

- Signal status for the process value PV_Out:
  The signal status of the output parameter PV_Out always corresponds to the signal status of input parameter PV or, if the block is in simulation mode, 16#60.
- Signal status for the setpoint value SP:
  The signal status of the SP output parameter is always equivalent to the signal status of input parameter SP_Ext or SP_Int, depending on how the setpoint is specified. If the internal setpoint SP_Int is used, the signal status is always output as 16#80.
- Signal status of the error signal ER:
  The signal status of output parameter ER is obtained from the worst signal status of the two output parameters PV_Out and SP and is output.
  The signal status 16#60 (external simulation) is suppressed because the block acts as a sink with external simulation.
  If the external error signal is activated (Feature.Bit14 = 1), the signal status of ER_Ext.ST is applied.
- Signal status for the manipulated variable MV:
  The signal status of output parameter MV is obtained in "automatic mode" or in "program mode" with default setpoint from the worst signal status of the two parameters FFwd and ER and is output. In "manual mode", the signal status is output as good. The signal status 16#60 (external simulation) is suppressed because the block acts as a sink with external simulation. In "manual mode", the signal status is output as good.

- Signal status for position feedback `RbkOut`:
  The signal status of `RbkOut` always corresponds to the signal status of input parameter `Rbk` or, if the block is in simulation mode, `16#60`.
- Worst signal status:
  The worst signal status `ST_Worst` for the block is formed from the following parameters:
  `Feature.Bit14` = 1 (external error signal)
  - `ER_Ext.ST`;
  - `FFwd.ST`;
  - `RbkOut.ST`;
  - `MV_ChnST.ST`;

  `Feature.Bit14` = 0 (external error signal)
  - `SP.ST`;
  - `PV_Out.ST`;
  - `FFwd.ST`;
  - `RbkOut.ST`;
  - `MV_ChnST.ST`;

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 4 | Setting switch or button mode (Page 177) |
| 5 | Changes signal status of outputs in OOS (Page 208) |
| 6 | Ramp rate calculation (Page 191) |
| 8 | Separate delay times for each alarm (Page 181) |
| 9 | Substitution value is active if the block is in bypass (Page 199) |
| 11 | Gradient limitation with time duration (Page 197) |
| 12 | Control zone with specified I component (Page 171) |
| 13 | Control zone with frozen I component (Page 171) |
| 14 | External control deviation (Page 161) |
| 15 | Neutral position manipulated variable takes effect with "out of service" operating mode (Page 177) |
| 16 | Neutral position manipulated variable takes effect at startup (Page 176) |
| 18 | Disabling bumpless switchover to automatic mode for controllers (Page 183) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 23 | SP following PV in open loop has no priority over SP_Ext and SP limits (Page 193) |
| 24 | Enabling local operator authorization (Page 169) |

| Bit | Function |
|---|---|
| 25 | Suppression of all messages (Page 185) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 28 | Disabling operating points (Page 156) |
| 29 | Signaling limit violation (Page 181) |
| 30 | Dead band is temporarily disabled (Page 152) |
| 31 | Limits disabled in faceplate (Page 206) |

### Configurable reactions using the `Feature2` parameter

| Bit | Function |
|---|---|
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) |
| 11 | Suppress MsgLock and “Out of service” mode for a connected message block (Page 202) |
| 12 | Disable limit act output when limit value message is disabled (Page 209) |

### Operator permissions

The block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" `AutModOp` |
| 1 | 1 = Operator can switch to "manual mode" `ManModOp` |
| 2 | 1 = Operator can switch to "Out of service" mode `OosOp` |
| 3 | 1 = Operator can switch to "program mode" `AdvCoEn` |
| 4 | 1 = Operator can switch the setpoint to "external" `SP_ExtOp` |
| 5 | 1 = Operator can switch the setpoint to "internal" `SP_IntOp` |
| 6 | 1 = Operator can change the internal setpoint `SP_Int` |
| 7 | 1 = Operator can change the manual parameter `Man` |
| 8 | 1 = Operator can change operation high limit of the setpoint `SP_InHiLim` |
| 9 | 1 = Operator can change operation low limit of the setpoint `SP_InLoLim` |
| 10 | 1 = Operator can change the operation high limit of the manipulated variable `ManHiLim` |
| 11 | 1 = Operator can change the operation low limit of the manipulated variable `ManLoLim` |
| 12 | 1 = Operator can enable the setpoint's gradient limitation function `SP_RateOn` |
| 13 | 1 = Operator can change the setpoint's high limit for the ramp `SP_UpRaLim` |
| 14 | 1 = Operator can change the setpoint's low limit for the ramp `SP_DnRaLim` |
| 15 | 1 = Operator can switch between the time value or the value for the ramp `SP_RmpModTime` |
| 16 | 1 = Operator can change the ramp time `SP_RmpTime` |
| 17 | 1 = Operator can change the target setpoint `SP_RmpTarget` for the setpoint ramp |
| 18 | 1 = Operator can enable the setpoint ramp function `SP_RmpOn` |
| 19 | 1 = Operator can permit the PID optimization function `OptimEn` |
| 20 | 1 = Operator can enable the track setpoint in "manual mode" function `SP_TrkPV` |

| Bit | Function |
|---|---|
| 21 | 1 = Operator can enable the bumpless switchover from external to internal `SP_TrkExt` |
| 22 | 1 = Operator can change the gain parameter `Gain` |
| 23 | 1 = Operator can change the integral time parameter `TI` |
| 24 | 1 = Operator can change the derivative time parameter `TD` |
| 25 | 1 = Operator can change the derivative gain parameter `DiffGain` |
| 26 | 1 = Operator can change the dead band parameter `DeadBand` |
| 27 | 1 = Operator can change the control zone parameter `ConZone` |
| 28 | 1 = Operator can change the derivative gain parameter `ER_AH_DFac` |
| 29 | 1 = Operator can change the derivative gain parameter `ER_AL_DFac` |
| 30 | Not used |
| 31 | Not used |

The block has the following permissions for the `OS1Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can change the limit (process value) `PV_AH_Lim` for the high alarm |
| 1 | 1 = Operator can change the limit (process value) `PV_WH_Lim` for the high warning |
| 2 | 1 = Operator can change the limit (process value) `PV_TH_Lim` for the high tolerance |
| 3 | 1 = Operator can change the hysteresis (process value) `PV_Hyst` |
| 4 | 1 = Operator can change the limit (process value) `PV_TL_Lim` for the low tolerance |
| 5 | 1 = Operator can change the limit (process value) `PV_WL_Lim` for the low warning |
| 6 | 1 = Operator can change the limit (process value) `PV_AL_Lim` for the low alarm |
| 7 | 1 = Operator can change the limit (error signal) `ER_AH_Lim` for the high alarm |
| 8 | 1 = Operator can change the hysteresis (error signal) `ER_Hyst` |
| 9 | 1 = Operator can change the limit (error signal) `ER_AL_Lim` for the low alarm |
| 10 | 1 = Operator can change the limit (position feedback) `RbkWH_Lim` for the high warning |
| 11 | 1 = Operator can change the hysteresis (position feedback) `RbkHyst` |
| 12 | 1 = Operator can change the limit (position feedback) `RbkWL_Lim` for the low warning |
| 13 | Not Used |
| 14 | 1 = Operator can activate bypass functionality |
| 15 | 1 = Operator can deactivate bypass functionality |
| 16 | 1 = Operator can activate the Simulation function `SimOn` |
| 17 | 1 = Operator can activate the Release for maintenance function `MS_RelOp` |
| 18 | 1 = Operator can activate / deactivate messages via `PV_AH_MsgEn` |
| 19 | 1 = Operator can activate / deactivate messages via `PV_WH_MsgEn` |
| 20 | 1 = Operator can activate / deactivate messages via `PV_TH_MsgEn` |
| 21 | 1 = Operator can activate / deactivate messages via `PV_TL_MsgEn` |
| 22 | 1 = Operator can activate / deactivate messages via `PV_WL_MsgEn` |
| 23 | 1 = Operator can activate / deactivate messages via `PV_AL_MsgEn` |
| 24 | 1 = Operator can activate / deactivate messages via `ER_AH_MsgEn` |
| 25 | 1 = Operator can activate / deactivate messages via `ER_AL_MsgEn` |
| 26 | 1 = Operator can activate / deactivate messages via `RbkWH_MsgEn` |
| 27 | 1 = Operator can activate / deactivate messages via `RbkWL_MsgEn` |

| Bit | Function |
|---|---|
| 28 | 1 = Operator can change the simulation value SimPV |
| 29 | 1 = Operator can change the simulation value SimRbk |
| 30 | 1 = Operator can activate the derivative action to the feedback path DiffToFbk |
| 31 | 1 = Operator can change the proportional action to the feedback path PropFacSP |

The block has the following permissions for the OS2Perm parameter:

| Bit | Function |
|---|---|
| 0 - 15 | reserved |

**Note**

If you interconnect a parameter that is also listed in OS_Perm as a parameter, you have to reset the corresponding OS_Perm bit.

## Interlocks

This block provides the following interlocks:

- Interlock without reset ("Interlock")

If interlock is active without forcing (MV_ForOn =0) the manipulated value MV.Value is set to the neutral position value ( Neutral position for motors, valves and controllers (Page 54) ).

You can find additional information on this in the section Interlocks (Page 107).

## Disabling interlocks

This block provides the standard function Disabling interlocks (Page 111).

## Release for maintenance

The block provides the standard function Release for maintenance (Page 70)

## Generating instance-specific messages

The block provides the standard function Generating instance-specific messages (Page 230) without the time stamp function in the I/O.

## Suppressing messages using the MsgLock parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

## Button labels

This block provides the standard function Labeling of buttons and text (Page 235).

Instance-specific text can be configured for the following parameters:

- AutModOp
- ManModOp
- AdvCoOn
- OosOp
- SP_ExtOp
- SP_IntOp

### Specifying the display area for process and setpoint values as well as operations

This block provides the standard function Display and operator input area for process values and setpoints (Page 233).

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### See also

PIDConL I/Os (Page 1039)

PIDConL block diagram (Page 1058)

PIDConL error handling (Page 1034)

PIDConL modes (Page 1022)

EventTs functions (Page 2046)

## 6.7.4 PIDConL error handling

### Error handling of PIDConL

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Control system fault (CSF)

## Overview of error numbers

The ErrorNum output parameter can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. |
| 30 | The value of PV can no longer be displayed in the REAL number field. |
| 31 | The value of SP_Ext can no longer be displayed in the REAL number field. |
| 32 | The value of FFwd can no longer be displayed in the REAL number field. |
| 33 | The value of MV_Trk can no longer be displayed in the REAL number field. |
| 34 | The value of MV_Forced can no longer be displayed in the REAL number field. |
| 35 | The value of Rbk can no longer be displayed in the REAL number field. |
| 36 | The value of MV can no longer be displayed in the REAL number field. |
| 51 | AutModLi = 1 and ManModLi = 1<br>SP_LiOp = 1 and SP_IntLi = 1 and SP_ExtLi = 1 |
| 52 | Feature.Bit30 = 1 and MeanMV.ST = 16#78 |
| 59 | = 1, "Gain is negative" |
| 60 | \|TI\| < SampleTime / 2 |
| 61 | \|TD\| < SampleTime |
| 62 | DiffGain < 1 or DiffGain > 10 |
| 63 | TD / DiffGain < SampleTime / 2 |
| 64 | PropFacSP < 0 or PropFacSP > 1 |
| 66 | NormPV_High = NormPV_Low |
| 74 | ConZone < 0.0 |

## Control system fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 128) section for more on this.

## See also

PIDConL block diagram (Page 1058)

PIDConL I/Os (Page 1039)

PIDConL messaging (Page 1036)

PIDConL functions (Page 1024)

PIDConL modes (Page 1022)

Description of PIDConL (Page 1017)
Setting switch or button mode (Page 177)

## 6.7.5 PIDConL messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Process messages
- Instance-specific messages

### Process control fault

The following control system error messages can be output:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId2` | SIG 6 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter `CSF`. If it changes to `CSF` = 1, a process control fault is triggered (`MsgEvId2`, SIG 6).

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 1 | Alarm - high | $$BlockComment$$<br>PV - high alarm limit violated |
| | SIG 2 | Warning - high | $$BlockComment$$<br>PV - high warning limit violated |
| | SIG 3 | Tolerance - high | $$BlockComment$$<br>PV - high tolerance limit violated |
| | SIG 4 | Tolerance - low | $$BlockComment$$<br>PV - low tolerance limit violated |
| | SIG 5 | Warning - low | $$BlockComment$$<br>PV - low warning limit violated |
| | SIG 6 | Alarm - low | $$BlockComment$$<br>PV - low alarm limit violated |
| | SIG 7 | Alarm - high | $$BlockComment$$<br>ER - high alarm limit violated |
| | SIG 8 | Alarm - low | $$BlockComment$$<br>ER - low alarm limit violated |
| MsgEvId2 | SIG 7 | Warning - high | $$BlockComment$$<br>Rbk - high warning limit violated |
| | SIG 8 | Warning - low | $$BlockComment$$<br>Rbk - low warning limit violated |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Instance-specific messages

You can use up to four instance-specific messages with this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId2 | SIG 1 | AS process control message - fault | $$BlockComment$$<br>External message 1<br>Status 16#@5%x@ |
| | SIG 2 | AS process control message - fault | $$BlockComment$$<br>External message 2<br>Status 16#@6%x@ |
| | SIG 3 | AS process control message - fault | $$BlockComment$$<br>External message 3<br>Status 16#@7%x@ |
| | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External message 4<br>Status 16#@8%x@ |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance **MsgEvId1**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |
| 4 | Process value PV_Out |
| 5 | Control deviation ER |
| 6 | ExtVa106 |
| 7 | ExtVa107 |
| 8 | Not allocated |
| 9 | Not allocated |
| 10 | Not allocated |

The associated values 6 ... 7 are allocated to the parameters ExtVa106 ... ExtVa107 and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### Associated values for message instance **MsgEvId2**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |
| 4 | Position feedback RbkOut |
| 5 | Signal status ExtMsg1 |
| 6 | Signal status ExtMsg2 |
| 7 | Signal status ExtMsg3 |
| 8 | Signal status ExtMsg4 |
| 9 | ExtVa209 |
| 10 | ExtVa210 |

The associated values 9 ... 10 are allocated to the parameters ExtVa209 ... ExtVa210 and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of PIDConL (Page 1017)

PIDConL functions (Page 1024)

PIDConL I/Os (Page 1039)

PIDConL block diagram (Page 1058)

PIDConL error handling (Page 1034)

PIDConL modes (Page 1022)

## 6.7.6 PIDConL I/Os

I/Os of PIDConL

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AdvCoEn | 1 = Enable "program mode" via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AdvCoOn* | 1 = Enable "program mode" via faceplate.<br>The parameter is available on the WinCC OPC server to be read by external APC programs. | BOOL | 0 |
| AdvCoModSP | Type of "program mode":<br>1 = Setpoint specification<br>0 = Manipulated variable specification | BOOL | 1 |
| AdvCoMstrOn | Activate (0-1) or deactivate (1-0) "program mode" via edge transition | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AdvCoMV | Specified value from the external program.<br>The parameter is available on the WinCC OPC server to be read by external APC programs. | REAL | 0.0 |
| AutModLi* | 1 = "Automatic mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutModOp* | 1 = "Automatic mode" via operator (controlled by ModLiOp = 0) | BOOL | 0 |
| BatchEn | 1 = Enable allocation for batch control | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkTypeData* | This input can not be edit or linked and is for internal usage only! | | |
| BypProt | 1 = Interlock bypass during simulation | BOOL | 0 |
| BypLiOp | 1 = Bypass commands via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypLock | 1 = Bypass activation or deactivation is locked for operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypPV | Substitution value if block is in bypass | REAL | 0.0 |
| BypPVLi | 1 = Select bypass PV (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypPVOp | 1 = Select bypass PV (via operator) | BOOL | 0 |
| ConZone | Width of control zone | REAL | 0.0 |
| CPI_In | Input for control performance index, which is calculated by the assigned ConPerMon block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#78 |
| CSF | 1 = External error (control system fault) Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DeadBand | Width of dead band | REAL | 0.0 |
| DiffGain | Gain of differentiator [1..10] DiffGain = TD / (delay time of D component) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 5.0<br>• 16#80 |
| DiffSel | 1 = D component activated | BOOL | 1 |
| DiffToFbk | 1 = D component is placed in the feedback | BOOL | 0 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| ER_A_DC* | Delay time for incoming ER high/low or only low alarms [s] | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ER_AH_DC* | Delay time for incoming ER high alarms [s] | REAL | 0.0 |
| ER_A_DG* | Delay time for outgoing ER high/low or only low alarms [s] | REAL | 0.0 |
| ER_AH_DG* | Delay time for outgoing ER high alarms [s] | REAL | 0.0 |
| ER_AH_En | 1 = Activate alarm (high) for error signal monitoring | BOOL | 1 |
| ER_AH_DFac* | Delay factor at positive setpoint step changes for incoming alarms at the error signal monitoring ER_AH_Lim | REAL | 0.0 |
| ER_AH_Lim | Alarm limit (high) for error signal monitoring | REAL | 100.0 |
| ER_AH_MsgEn | 1 = Activate messages for alarm (high) for error signal monitoring | BOOL | 1 |
| ER_AL_DFac* | Delay factor at negative setpoint step changes for incoming alarms at the error signal monitoring ER_AL_Lim | REAL | 0.0 |
| ER_AL_En | 1 = Activate alarm (low) for error signal monitoring | BOOL | 1 |
| ER_AL_Lim | Alarm limit (low) for error signal monitoring | REAL | -100.0 |
| ER_AL_MsgEn | 1 = Activate messages for alarm (low) for error signal monitoring | BOOL | 1 |
| ER_Ext | External error signal | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ER_Hyst | Alarm hysteresis for error signal | REAL | 1.0 |
| EventTsIn | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| ExtMsg1 | 1 = Binary input for freely selectable message 1 is used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | 1 = Binary input for freely selectable message 2 is used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ExtMsg3 | 1 = Binary input for freely selectable message 3 is used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg4 | 1 = Binary input for freely selectable message 4 is used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| ExtVa209 | Associated value 9 for messages (MsgEvID2) | ANY | |
| ExtVa210 | Associated value 10 for messages (MsgEvID2) | ANY | |
| Feature | I/O for additional functions (Page 1024) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| Feature2 | I/O for additional functions (Page 1024) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| FFwd* | Input for additive disturbance variable activation | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| FFwdHiLim | Limit (high) for additive disturbance variable activation | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| FFwdLoLim | Limit (low) for additive disturbance variable activation | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• -100.0<br>• 16#80 |
| Gain | Proportional gain<br>Gain.ST = 16#FF: Enabled in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#FF |
| Intl_En | 1 = Interlock without reset (interlock, Intlock parameter) is in effect | BOOL | 1 |
| IntHoldNeg | 1 = Integrator cannot run in negative direction | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| IntHoldPos | 1 = Integrator cannot run in positive direction | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Intlock | 0 = Interlock without reset is in effect<br>Once the interlock condition has cleared, the block can be operated without reset<br>1 = Interlock not activated | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| IntSel | 1 = I component activated | BOOL | 1 |
| Man* | Manual specification for the manipulated variable | REAL | 0.0 |
| ManHiLim | Limit (high) for manual parameter Man | REAL | 100.0 |
| ManLoLim | Limit (low) for manual parameter Man | REAL | 0.0 |
| ManModLi* | 1 = "Manual mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManModOp* | 1 = "Manual mode" via OS operator (controlled by ModLiOp = 0) | BOOL | 1 |
| ModLiOp | Operating mode switchover between:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_RelOp* | 1 = Release for maintenance by OS operator | BOOL | 0 |
| MsgEvID1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgEvID2 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Hotspot-Text (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_ChnST | Signal status of output channel for MV<br>Should be connected to an output channel block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| MV_Forced* | Forced manipulated variable that is not limited and assumes top priority | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_ForOn | 1 = Output forced manipulated variable MV_Forced unlimited at output MV | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_HiLim | Limit (high) for manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MV_LoLim | Limit (low) for manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_Mean | Mean value of the MV in the time window | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#78 |
| MV_Offset | Manipulated variable for ER=0, operating point for controller with deactivated I component | REAL | 0.0 |
| MV_OpScale | OS display range for manipulated variable MV | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| MV_SafePos | Manipulated variable neutral position | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_Trk* | Tracking value for the manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_TrkOn | 1 = Tracking of manipulated variable MV | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_Unit | Unit of measure for manipulated variable | INT | 1342 |
| NegGain | 0 = Effective proportional gain GainEff is positive<br>1 = Effective proportional gain GainEff is negative | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NormMV | Manipulated variable range (MV) for standardizing the proportional gain (GAIN) | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| NormPV | Process value range (PV) for standardizing the proportional gain (GAIN) | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OptimEn* | 1 = Enable optimization of PID parameters by PID tuner | BOOL | 0 |
| OptimOcc* | 1 = Optimization running | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 1024) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| OS1Perm | I/O for operator permissions (Page 1024) | STRUCT<br>• Bit 0: BOOL<br>• Bit 18: BOOL<br>• Bit 19: BOOL<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1<br>• 1 |
| OS2Perm | I/O for operator permissions (Page 1024) | STRUCT<br>• Bit 0: BOOL<br>• Bit 15: BOOL | -<br>• 1<br>• 1 |
| PropFacSP | Applying the P component to the feedback [0..1].<br>0 = P component fully in feedback | REAL | 1.0 |
| PropSel | 1 = Activate P component | BOOL | 1 |
| PV* | Process value (controlled variable) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_A_DC* | Delay time for incoming PV high/low or only low alarms [s] | REAL | 0.0 |
| PV_AH_DC* | Delay time for incoming PV high alarms [s] | REAL | 0.0 |
| PV_A_DG* | Delay time for outgoing PV high/low or only low alarms [s] | REAL | 0.0 |
| PV_AH_DG* | Delay time for outgoing PV high alarms [s] | REAL | 0.0 |
| PV_AH_En | 1 = Enable PV alarm limit (high) | BOOL | 1 |
| PV_AH_Lim | Limit PV alarm (high) | REAL | 95.0 |
| PV_AH_MsgEn | 1 = Enable PV alarm (high) message | BOOL | 1 |
| PV_AL_En | 1 = Enable PV alarm limit (low) | BOOL | 1 |
| PV_AL_Lim | PV alarm limit (low) | REAL | 5.0 |
| PV_AL_MsgEn | 1 = Enable PV alarm (low) message | BOOL | 1 |
| PV_Hyst | Hysteresis for PV alarm, warning and tolerance limits | REAL | 1.0 |
| PV_OpScale | Limit for scale in PV bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| PV_T_DC* | Delay time for incoming PV high/low or only low tolerance messages [s] | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_TH_DC* | Delay time for incoming PV high tolerance messages [s] | REAL | 0.0 |
| PV_T_DG* | Delay time for outgoing PV high/low or only low tolerance messages [s] | REAL | 0.0 |
| PV_TH_DG* | Delay time for outgoing PV high tolerance messages [s] | REAL | 0.0 |
| PV_TH_En | 1 = Enable PV tolerance limit (high) | BOOL | 0 |
| PV_TH_Lim | Limit PV tolerance message (high) | REAL | 85.0 |
| PV_TH_MsgEn | 1 = Enable message for PV tolerance message (high) | BOOL | 1 |
| PV_TL_En | 1 = Enable PV tolerance limit (low) | BOOL | 0 |
| PV_TL_Lim | Limit PV tolerance message (low) | REAL | 15.0 |
| PV_TL_MsgEn | 1 = Enable message for tolerance message (low) | BOOL | 1 |
| PV_Unit | Unit of measure for process value | INT | 1001 |
| PV_W_DC* | Delay time for incoming PV high/low or only low warnings [s] | REAL | 0.0 |
| PV_WH_DC* | Delay time for incoming PV high warnings [s] | REAL | 0.0 |
| PV_W_DG* | Delay time for outgoing PV high/low or only low warnings [s] | REAL | 0.0 |
| PV_WH_DG* | Delay time for outgoing PV high warnings [s] | REAL | 0.0 |
| PV_WH_En | 1 = Enable PV warning limit (high) | BOOL | 1 |
| PV_WH_Lim | Limit PV warning (high) | REAL | 90.0 |
| PV_WH_MsgEn | 1 = Enable PV warning (high) message | BOOL | 1 |
| PV_WL_En | 1 = Enable PV warning limit (low) | BOOL | 1 |
| PV_WL_Lim | Limit PV warning (low) | REAL | 10.0 |
| PV_WL_MsgEn | 1 = Enable PV warning (low) message | BOOL | 1 |
| Rbk* | Position feedback for display on OS | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| RbkHyst | Alarm hysteresis for position feedback | REAL | 1.0 |
| RbkWH_En | 1 = Enable warning (high) for position feedback | BOOL | 1 |
| RbkWH_Lim | Limit for position feedback of warning (high) | REAL | 100.0 |
| RbkWH_MsgEn | 1 = Enable messages for warning (high) for position feedback | BOOL | 1 |
| RbkWL_En | 1 = Enable warning (low) for position feedback | BOOL | 1 |
| RbkWL_Lim | Limit for position feedback of warning (low) | REAL | 0.0 |
| RbkWL_MsgEn | 1 = Enable messages for warning (low) for position feedback | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RefStdDevIn | Reference value of PV standard deviation (sigma) in defined "good" state of control loop | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#78 |
| RstBypLi | 1 = Reset bypass PV (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstBypOp | 1 = Reset bypass PV (via operator) | BOOL | 0 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SafePos | 1 = Neutral position (Page 54) for controller manipulated variable is ManHiLim<br>0 = Neutral position for controller manipulated variable is ManLoLim | BOOL | 0 |
| SafePos2 | Neutral position for controller manipulated variable:<br>0 = SafePos is valid<br>1 = Neutral position is MV_SafePos<br>2 = Neutral position is last manipulated variable (stop) | INT | 0 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | 1 = Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SelFp2 | 1 = Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | - |
| SettliTime | Settling time [s] of the control loop determined by the ConPerMon block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#78 |
| SettliFactor | Factor to increase the settling time to adjust the dead band | REAL | 2.0 |
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Vale: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SimPV* | Process value used for SimOn = 1 | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SimPV_Li | Process value that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SimRbk* | Position feedback used for SimOn = 1 | REAL | 0.0 |
| SimRbkLi | Position feedback used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_DnRaLim* | Limit (low) for the gradient of the setpoint [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 100.0 |
| SP_ExHiLim | Limit (high) for external setpoint | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| SP_ExLoLim | Limit (low) for external setpoint | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_Ext* | external setpoint - (to interconnection) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_ExtLi* | 1 = Select external setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtOp* | 1 = Select external setpoint (via operator) | BOOL | 0 |
| SP_InHiLim | Limit (high) of internal setpoint | REAL | 100.0 |
| SP_InLoLim | Limit (low) of internal setpoint | REAL | 0.0 |
| SP_Int* | Internal setpoint for operation | REAL | 0.0 |
| SP_IntLi* | 1 = Select internal setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_IntOp* | 1 = Select internal setpoint (via operator) | BOOL | 0 |
| SP_LiDnRaLim | Linkable limit (low) for the gradient of the setpoint [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_LiOp | Select setpoint source (internal/external):<br>1 = Via interconnection<br>0 = Via operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_LiRmpTime | Linkable time [Time unit] for gradient limitation in ramp mode duration<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_LiUpRaLim | Linkable gradient limit (high) for the setpoint [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP_RateOn* | 1 = Activate limitation of setpoint gradients | BOOL | 0 |
| SP_RmpModTime | 1 = Use time values (SP_RmpTime, SP_LiRmpTime)<br>0 = Use gradient limits (SP_DnRaLim, SP_LiDnRaLim, SP_UpRaLim, SP_LiUpRaLim) | BOOL | 0 |
| SP_RmpOn* | 1 = Activate setpoint ramp to target setpoint SP_RmpTarget | BOOL | 0 |
| SP_RmpTarget* | Target setpoint for setpoint ramp | REAL | 0.0 |
| SP_RmpTime* | Time [Time unit] for setpoint ramp from current SP up to SP_RmpTarget or time [Time unit] for gradient limitation in ramp mode duration.<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_TrkExt | 1 = Bumpless switchover from external to internal setpoint active | BOOL | 1 |
| SP_TrkPV | 1 = Setpoint follows PV in "manual mode" and with tracking | BOOL | 0 |
| SP_UpRaLim* | Gradient limit (high) for the setpoint [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 100.0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| TD | Derivative time [s]<br>TD.ST = 16#FF: Enabled in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| TI | Integral time [s]<br>TI.ST = 16#FF: Enabled in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#FF |
| TimeFactor | Time unit:<br>0 = Seconds<br>1 = Minutes<br>2 = Hours | INT | 0 |
| UA1unit | Unit of measure for analog auxiliary value 1 | INT | 0 |
| UA2unit | Unit of measure for analog auxiliary value 2 | INT | 0 |
| UserAna1 | Analog auxiliary value 1 | STRUCT<br>Value: REAL<br>ST: BYTE | <br>0.0<br>16#FF |
| UserAna2 | Analog auxiliary value 2 | STRUCT<br>Value: REAL<br>ST: BYTE | <br>0.0<br>16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AdvCoAct | 1 = "Program mode" enabled.<br>The parameter is available on the WinCC OPC server to be read by external APC programs. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AdvCoRdy | 1 = "Program mode" available | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutAct | 1 = "Automatic mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypassAct | 1 = Bypass is activated in this block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CascaCut | Cascade connection:<br>1 = Control chain from primary to secondary controller is interrupted | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| DynDeadBand | Dynamic dead band | REAL | 0.0 |
| D_Part | D component of PID algorithm | REAL | 0.0 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ER | Error signal | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ER_A_DCOut | Effective delay time [s] for incoming alarms at the error signal monitoring | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ER_AH_Act | 1 = Alarm limit (high) for control deviation violated.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| ER_AL_Act | 1 = Alarm limit (low) for control deviation violated.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| ER_AL_LimOut | ER_AL_Lim as output | REAL | 0.0 |
| ER_AH_LimOut | ER_AH_Lim as output | REAL | 0.0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see PIDConL error handling (Page 1034) | INT | -1 |
| FFwdHiAct | 1 = Limit (high) for additive disturbance variable activation violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FFwdLoAct | 1 = Limit (low) for additive disturbance variable activation violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| GainEff | Effective proportional gain depends on NegGain, Gain, NormPV, and NormMV | REAL | 1.0 |
| GrpErr | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IntlockOut | Intlock as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| I_Part | I component of PID algorithm | REAL | 0.0 |
| LockAct | 1 = Interlock is in effect | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LoopClosed | 1 = Control loop closed<br>0 = Control loop open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManAct | 1 = "Manual mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ManHiOut | Limit (high) for "manual mode", corresponds to input parameter ManHiLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| ManLoOut | Limit (low) for "manual mode", corresponds to input parameter ManLoLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message acknowledgement status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |
| MsgAckn2 | Message acknowledgement status 2 (output ACK_STATE of second ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | 1 = Alarm error 1 (output ERROR of the first ALARM_8P) | BOOL | 0 |
| MsgErr2 | 1 = Alarm error 2 (output ERROR of the second ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| MsgStat2 | Message status 2 (output STATUS of second ALARM_8P) | WORD | 16#0000 |
| MV | Manipulated variable | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_HiAct | 1 = Limit (high) of manipulated variable violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_LoAct | 1 = Limit (low) of manipulated variable violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_UnitOut | Unit of measure for manipulated variable, for interconnecting to the MV_Unit input parameter of the ConPerMon block | INT | 0 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OS1PermOut | Display of OS1Perm | DWORD | 16#FFFFFFFF |
| OS1PermLog | Display of OS1Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS2PermOut | Display of OS2_Perm | WORD | 16#FFFF |
| OS2PermLog | Display of OS2_Perm with settings changed by the block algorithm | WORD | 16#FFFF |
| PhaseDeadBand | Phase for the dynamic adjustment of the dead band<br>0: Dead band enabled<br>1: Dead band disabled<br>2: Settling time | INT | 0 |
| P_Part | P component of PID algorithm | REAL | 0.0 |
| PV_AH_Act | 1 = PV alarm (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_AH_Out | PV - High alarm limit output | REAL | 0.0 |
| PV_AL_Act | 1 = PV alarm (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_AL_Out | PV - Low alarm limit output | REAL | 0.0 |
| PV_HysOut | PV - Alarm hysteresis output | REAL | 0.0 |
| PV_Out | Output for process value | STRUCT<br>• Value: REAL<br>• ST: BYTE<br>• Bit 1: BOOL of internal fill byte (Bypass information bit) | -<br>• 0.0<br>• 16#80<br>• 0 |
| PV_OpScaleOut | PV_OpScale as output | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| PV_TH_Act | 1 = PV tolerance message (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_TH_Out | PV - High tolerance limit output | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_TL_Act | 1 = PV tolerance message (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_TL_Out | PV - Low tolerance limit output | REAL | 0.0 |
| PV_ToleHi | Limit (high) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_ToleLo | Limit (low) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_WH_Act | 1 = PV warning (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_WH_Out | PV - High warning limit output | REAL | 0.0 |
| PV_WL_Act | 1 = PV warning (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_WL_Out | PV - Low warning limit output | REAL | 0.0 |
| PV_UnitOut | Unit of measure for process value, for interconnecting to the PV_Unit input parameter of the ConPerMon block | INT | 0 |
| RbkOut | Output for position feedback | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RbkWH_Act | 1 = Warning (high) for position feedback active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RbkWL_Act | 1 = Warning (low) for position feedback active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToStart | 1 = Active start readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SettlingTimer | Settling time for a closed control loop | REAL | 0.0 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP | Setpoint used by controller | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_DnRaAct | 1 = Negative gradient limiting of setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExHiAct | 1 = Limit (high) for external setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExLoAct | 1 = Limit (low) for external setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtAct | 1 = External setpoint is active<br>0 = Internal setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtOut | External setpoint, corresponds to input parameter SP_Ext | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_InHiOut | Limit (high) for SP_Int corresponds to input parameter SP_InHiLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| SP_InLoOut | Limit (low) for SP_Int corresponds to input parameter SP_InLoLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_RateTarget | Target setpoint for the gradient limitation | REAL | 0.0 |
| SP_UpRaAct | 1 = Positive gradient limiting of setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP_RemRT | Remaining ramp time of the setpoint | REAL | 0.0 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 1017) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1017) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1017) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 1017) | DWORD | 16#00000000 |
| SumMsgAct | 1 = Active hardware interrupt | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| UA1unitOut | UA1 as output | INT | 0 |
| UA2unitOut | UA2 as output | INT | 0 |
| UserAna1Out | UserAna1 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| UserAna2Out | UserAna2 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |

### Calculation of the output parameter ER_A_DCOut

ER_A_DC is assigned by default to the output before a setpoint change.

ER_A_DCOut = ER_A_DC

In the case of a setpoint change in the positive direction during automatic mode, the output is calculated as follows:

ER_A_DCOut = Maximum (ER_A_DC, ER_AH_DFac * Setpoint difference)

In the case of a setpoint change in the negative direction during automatic mode, the output is calculated as follows:

ER_A_DCOut = Maximum (ER_A_DC, -1*ER_AH_DFac * Setpoint difference)

When the control circuit has stabilized again, meaning

(ER_AL_Lim + ER_Hyst) <= ER <= (ER_AH_Lim – ER_Hyst)

and the delay time for outgoing alarms ER_A_DG has expired, the output is reset again to ER_A_DC: ER_A_DCOut = ER_A_DC

### Activating and deactivating the function:

The function is deactivated (default) when the following applies: ER_AH_DFac = 0.0 and ER_AL_DFac = 0.0

### See also

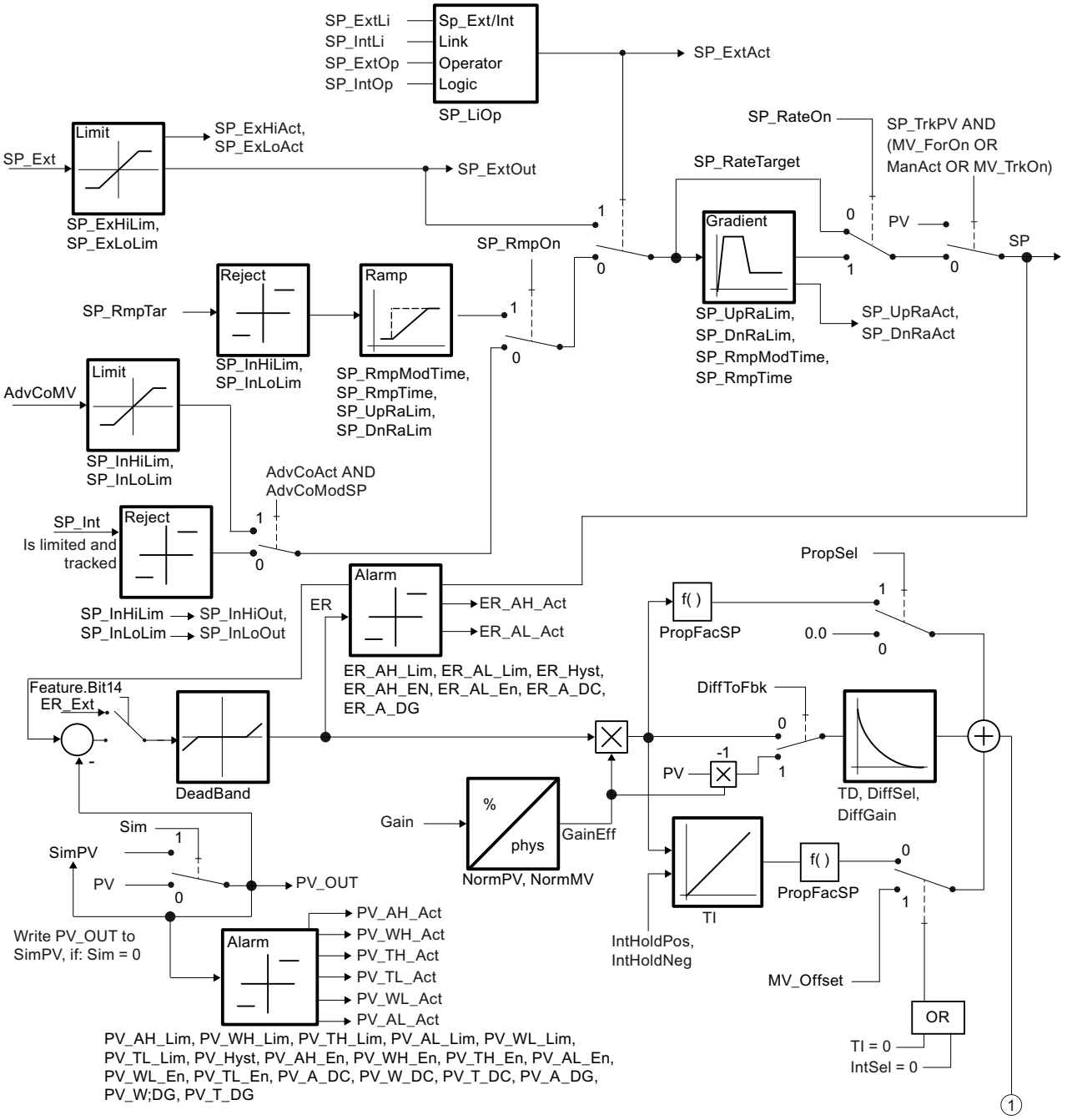
PIDConL messaging (Page 1036)

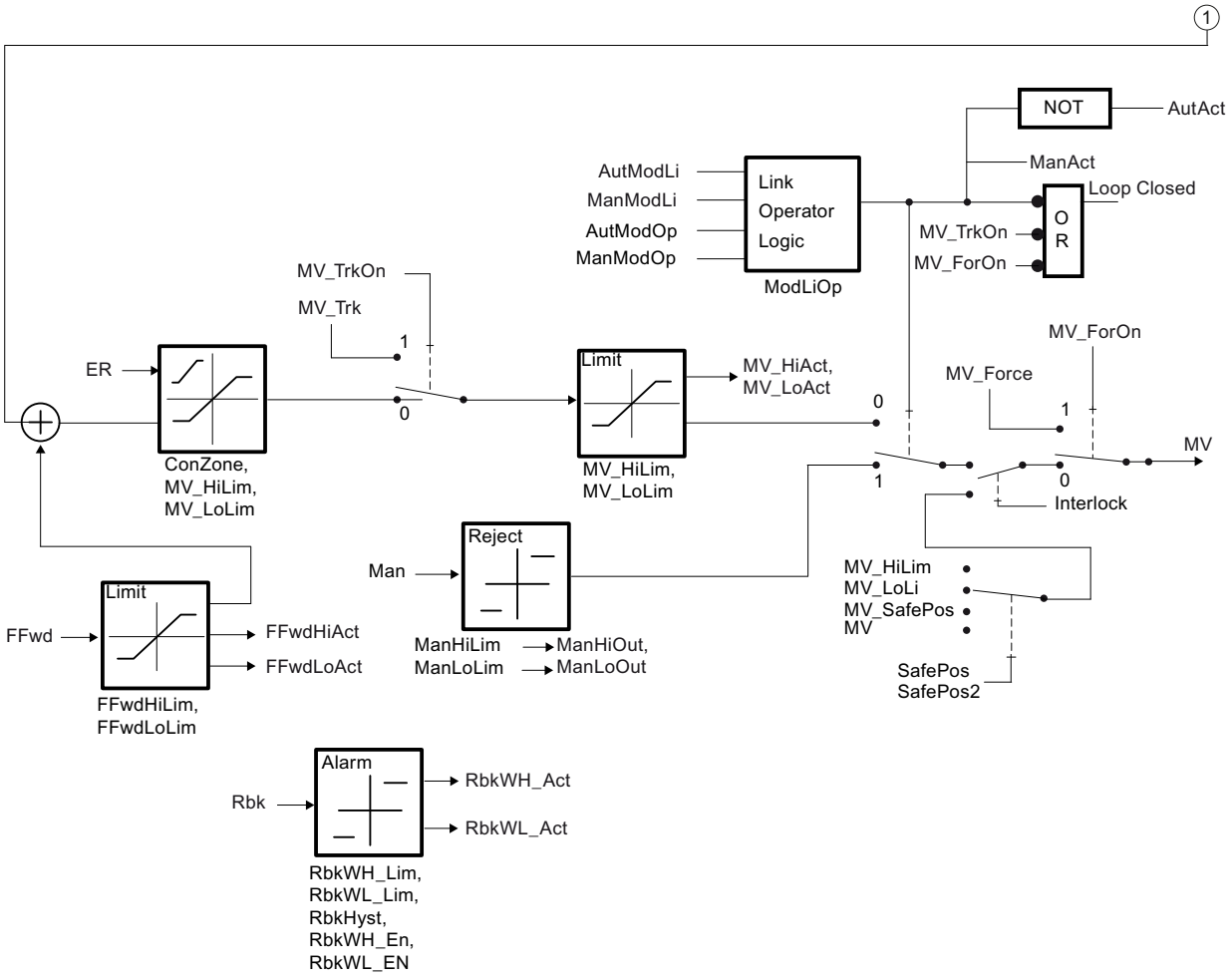
PIDConL block diagram (Page 1058)

PIDConL modes (Page 1022)

## 6.7.7 PIDConL block diagram

### PIDConL block diagram






## See also

PIDConL I/Os (Page 1039)

PIDConL messaging (Page 1036)

PIDConL error handling (Page 1034)

PIDConL functions (Page 1024)

PIDConL modes (Page 1022)

Description of PIDConL (Page 1017)

## 6.7.8 Operator control and monitoring

### 6.7.8.1 PIDConL views

#### Views of the PIDConL block

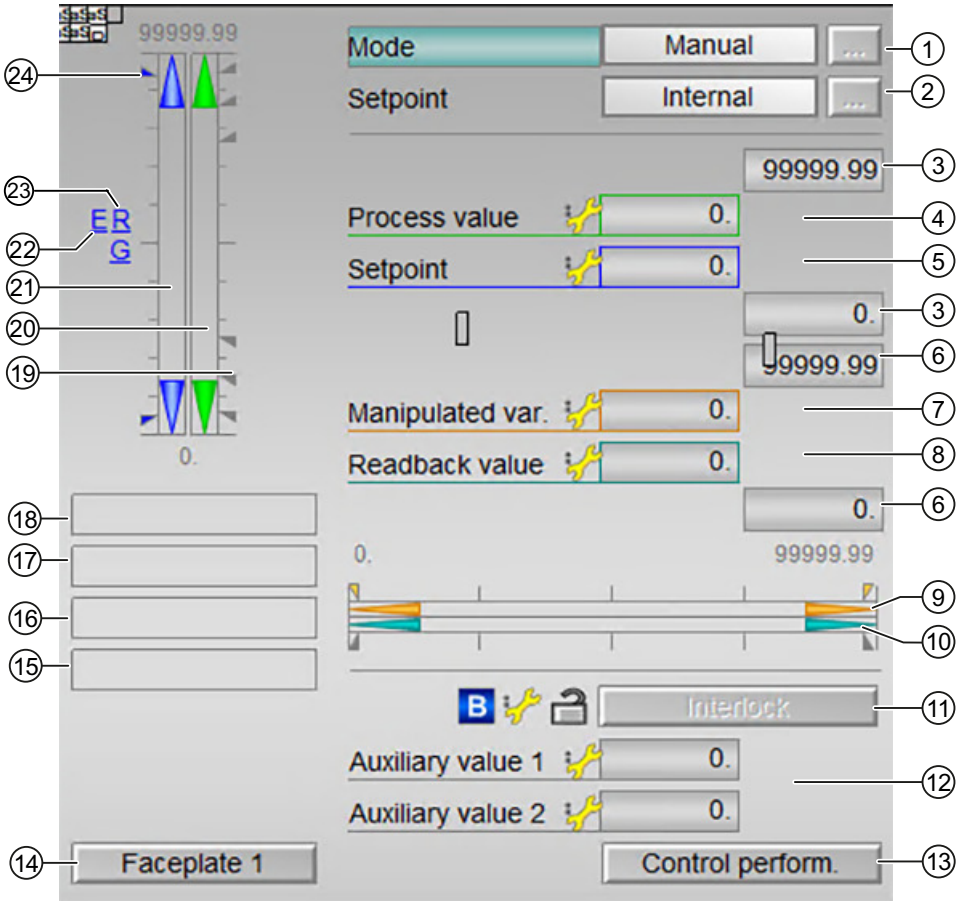
The block PIDConL provides the following views:

- PIDConL, PIDConS and PIDConR standard views (Page 1061)
- Alarm view (Page 336)
- Limit view of PID controllers (Page 325)
- Trend view (Page 341)
- Ramp view (Page 334)
- Parameter view of PID controllers (Page 315)
- PIDConL, PIDConS and PIDConR previews (Page 1066)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icons for PID and FM controller (Page 269)

Refer to the Structure of the faceplate (Page 279) and Block icon structure (Page 260) sections for general information about the faceplate and block icon.

### 6.7.8.2 PIDConL, PIDConS and PIDConR standard views

#### PIDConL standard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 79)
- Automatic mode (Page 79)
- Program mode for controllers (Page 85) (not with PIDConS)
- Out of service (Page 77)

Refer to section Switching operating states and operating modes (Page 289) for information on switching the operating mode.

### (2) Displaying and switching the setpoint

This area shows how to specify the setpoint. The setpoint can be specified as follows:

- By the application ("External", CFC/SFC)
- By the user directly in the faceplate ("Internal").

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the setpoint specification.

You can find additional information on this in the section Setpoint specification - internal/ external (Page 136).

**Note**

With the PIDConR block, this area is only visible if you have set the `Feature` Bit Switching operator controls for external setpoint to visible (Page 154) to 1.

### (3) High and low scale range for the process value

These values provide information on the display range for the bar graph of the process value. The scale range is defined in the engineering system. For more information refer Bar graph (Page 376) section.

### (4) Display of the process value including signal status

This area shows the current process value with the corresponding signal status.

### (5) Displaying and changing the setpoint including signal status

This area shows the current setpoint with the corresponding signal status.

Refer to the Changing values (Page 291) section for information on changing the setpoint. The setpoint specification also needs to be set to "Internal" for this block.

### (6) High and low scale range for the setpoint

This area is already set and cannot be changed.

### (7) Displaying and changing the manipulated variable including signal status

This area shows the current manipulated variable with the corresponding signal status.

Refer to the Changing values (Page 291) section for information on changing the manipulated variable. You can only make a change in manual mode.

### (8) Display of the position feedback including signal status

This display is only visible when the corresponding block input is connected.

This area shows the current feedback of the manipulated variable with the corresponding signal status.

### (9) Bar graph for the manipulated variable

This area shows the current manipulated variable in the form of a bar graph. You can also see the high and low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (10) Bar graph for position feedback

This display is only visible when the corresponding block input is connected.

This area shows the current position feedback in the form of a bar graph. You can also see the high and low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (11) Operator control and display area for interlock functions of the block (not with PIDConS)

This display is only visible when the corresponding block input is connected.

You can use this button to control the interlock functions of the block. You can find additional information on this in the section Interlocking functions (Page 107).

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 113)), for example:

- Signal status (see Forming and outputting the signal status for technologic blocks (Page 117)), for example:

- Bypass information (see Forming the group status for interlock information (Page 113)):

If the text is configured for the interlocks, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (12) Display for auxiliary values

You can use this area to display two auxiliary values that have been configured in the engineering system (ES). You can find more information on this in the section Displaying auxiliary values (Page 239).

**Note**

Auxiliary values are available only in `PIDConL` and `PIDConR`

### (13) Navigation button for the standard view of the ConPerMon block

Use this navigation button to open the standard view of the ConPerMon block. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See also the section Opening additional faceplates (Page 234) for more on this.

### (14) Navigation button for switching to the standard view of any faceplate

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

See also the section Opening additional faceplates (Page 234) for more on this.

### (15) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this is available in section Release for maintenance (Page 70) Display area for block states.

### (16) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"

You can find additional information on this in the section Simulating signals (Page 64).

### (17) Display area for block states

This area provides additional information on the operating state of the block:

- "Internal error"
- "Invalid Signal"

### (18) Display area for block states

This area provides additional information on the operating state of the block (from high to low according to priority):

- "Optimizing"
- "Tracking"
- "Forced tracking"
- "Load SP"
- "SP ramp active"

### (19) Limit display

These colored triangles indicate the configured limits in the respective bar graph:

- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

### (20) Bar graph for the process value

This area shows the current process value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (21) Bar graph for the setpoint

This area shows the current setpoint in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (22) Display of external setpoint

This display [E] is only visible when you have selected "Internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

### (23) Display for the target setpoint of the setpoint ramp

This display [R] shows the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 334).

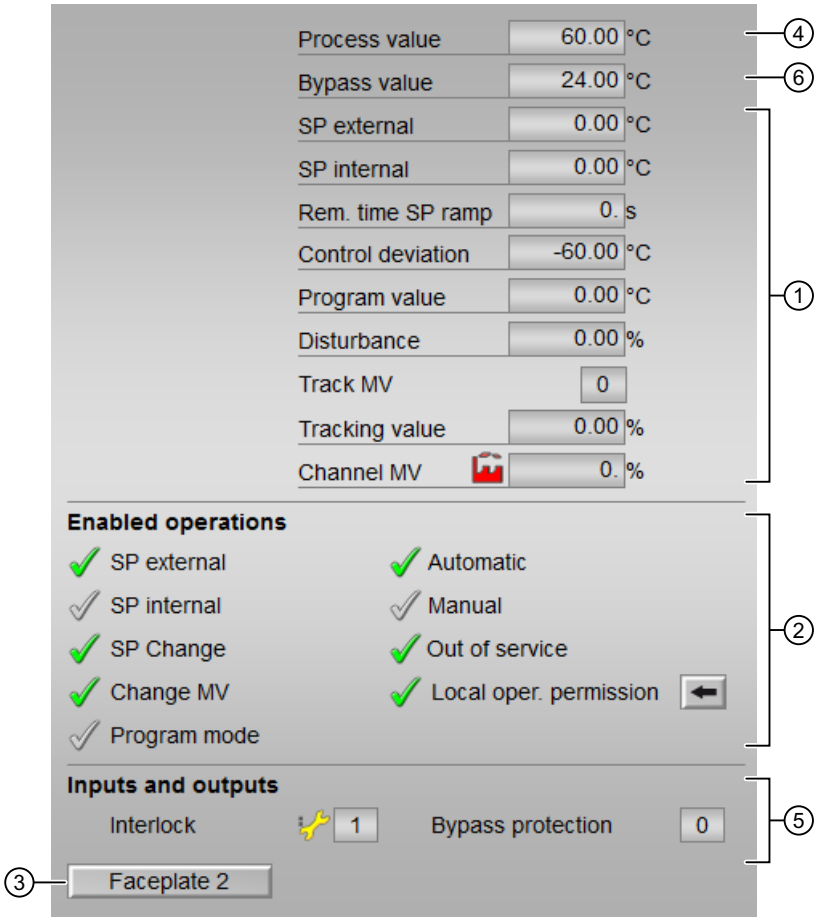
### (24) Limit display for the setpoint

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the engineering system (ES).

### 6.7.8.3 PIDConL, PIDConS and PIDConR previews

#### Preview of PIDConL

The preview shows you the parameters that you, as an OS operator, can control. You cannot control anything in this view, however.




#### (1) Preview area

This area shows you a preview for the following values:

- "SP external": currently applicable external setpoint
  - With the PIDConR block, this area is only visible if you have set the `Feature` Bit Switching operator controls for external setpoint to visible (Page 154) to 1
- "SP internal": currently applicable internal setpoint
- "Rem. time SP ramp" : Remaining time to reach the ramp target value (not with PIDConS)
- "Error signal": Current control deviation
- "Program value": Default value for program mode (not with PIDConS)
- "Disturbance variable": Additive value for feedforward control (not with PIDConS)

- "Track MV": Track manipulated variable (value is 1)
- "Tracking value": effective manipulated variable for "Track manipulated variable"
- "Channel MV": Display of the manipulated variable by the output channel block

### (2) Enable operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: The OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

The following enabled operations are shown here:

- "SP external": You can feedforward the external setpoint.
- "SP internal": You can feedforward the internal setpoint.
- "Change SP": You can change the setpoint.
- "Change MV": You can change the manipulated variable.
- "Program mode": You can switch to "program mode". (not with PIDConS)
- "Automatic": You can switch to "automatic mode".
- "Manual": You can switch to "manual mode".
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operator permission": Use the ← button to switch to the standard view of the OpStations block. You can find additional information on this in the section Operator control permissions (Page 285).

### (3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to open the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See also the section Opening additional faceplates (Page 234) for more on this.

### (4) Process value

This area displays the real process value (`PV`).

### (5) Inputs and outputs

This area shows the following parameters:

- "Interlock" (not with PIDConS):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock deact." (not with PIDConS):
  - 0 = Bypass disabled
  - 1 = Bypassing interlock in "local mode" and in "simulation"

### (6) Bypass value

This area displays the bypass value (`BypPV`).

# 6.8 PIDConS - Continuous PID controller (Small)

## 6.8.1 Description of PIDConS

### Object name (type + number) and family

Type and number: FB 1830

Family: Control

### Area of application for PIDConS

The block is used for the following applications:

- Fixed setpoint control

### How it works

The block is a PID controller with continuous output signal (manipulated variable). It is used to activate a final controlling element with continuous action input.

The block functions following the PID algorithm with a delayed D component and an integrator with double precision.

The block is suitable for controlling sluggish control loops, for example, for temperatures and filling levels, and high-speed control loops, for example, for flow rates and speed. For a given CPU, a compromise has to be made between the number of controllers and the frequency with which the individual controllers have to be processed. The faster the modulated control loops are, i.e. the more frequently the manipulated variables have to be calculated per time unit, the lower the number of controllers that can be installed.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

For the control loop monitoring to work as planned in the trend view of the controller faceplates, the

```
S7_xarchive:='Value, shortterm;'
```

attributes in the process tag types for control loops at the controller function block must be set for the following tags:

- Input parameters:
  - `CPI_In`
- Output parameters
  - `MV`
  - `MV_HiAct`
  - `MV_LoAct`
  - `LoopClosed`
  - `SP`
  - `PV_Out`
  - `PV_ToleHi`
  - `PV_ToleLo`

There are no templates for the PIDConS block for process tag types or simulated use cases in the example project (APL_Example_xx, xx represents the language variant). However, you can replace PIDConL with PIDConS in the following process tag types and use cases.

Examples of process tag types:

- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 2822)

Application cases in example project:

- Process simulation including noise generator (ProcSimC; ProcSimS) (Page 2871)
- Control loop monitoring for simulation with colored noise (ConPerMonSim) (Page 2875)
- Operating point-oriented adaptation of parameters (gain scheduling) for non-linear processes (GainSchedSim) (Page 2876)
- Filtering of noisy measured values in a control loop (SigSmoothSim) (Page 2877)

### Startup characteristics

Use the `Feature` bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section PIDConS I/Os (Page 1082).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |

| Status bit | Parameter |
|---|---|
| 3 | OosAct.Value |
| 4 | OosLi.Value |
| 5 | AutAct.Value |
| 6 | Invalid signal status |
| 7 | ManAct.Value |
| 8 | SP_ExtAct.Value |
| 9 | 1 = Input parameter MV_ChnST is interconnected |
| 10 | MV_TrkOn.Value AND NOT (ManAct.Value OR OosAct.Value) |
| 11 | MV.Value > MV_LoLim.Value |
| 12 - 17 | Not used |
| 18 | SimOn AND ManAct |
| 19 | Not used |
| 20 | 1 = Input parameter Rbk is not interconnected (Rbk.ST = 16#FF) |
| 21-22 | Not used |
| 23 | OptimEn |
| 24 | OptimOcc |
| 25 - 30 | Not used |
| 31 | Internal error |

## Status word allocation for Status2 parameter

| Status bit | Parameter |
|---|---|
| 0 | MsgLock |
| 1 | PV_AH_Act.Value |
| 2 | PV_WH_Act.Value |
| 3-4 | Not used |
| 5 | PV_WL_Act.Value |
| 6 | PV_AL_Act.Value |
| 7 | PV_AH_En OR Feature.Bit31 |
| 8 | PV_WH_En OR Feature.Bit31 |
| 9-10 | Not used |
| 11 | PV_WL_En OR Feature.Bit31 |
| 12 | PV_AL_En OR Feature.Bit31 |
| 13 | PV_AH_MsgEn |
| 14 | PV_WH_MsgEn |
| 15-16 | Not used |
| 17 | PV_WL_MsgEn |
| 18 | PV_AL_MsgEn |
| 19 | Delay of the PV_AH_Lim message |
| 20 | Delay of the PV_WH_Lim message |
| 21 | Delay of the PV_WL_Lim message |
| 22 | Delay of the PV_AL_Lim message |

| Status bit | Parameter |
|---|---|
| 23 | Collection of message delays |
| 24 | 1= Suppressed messages |
| 25-28 | Not used |
| 29 | `GrpErr.Value` |
| 30 | `RdyToStart.Value` |
| 31 | `MS_RelOp` |

**See also**

PIDConS modes (Page 1072)

PIDConS functions (Page 1073)

PIDConS error handling (Page 1079)

PIDConS messaging (Page 1080)

PIDConS block diagram (Page 1091)

## 6.8.2 PIDConS modes

### PIDConS operating modes

The block can be operated using the following modes:

- Automatic mode (Page 79)
- Manual mode (Page 79)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and bumpless switchover in the Manual and automatic mode for control blocks (Page 79) chapter.

### "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the Manual and automatic mode for control blocks (Page 79) chapter.

### "Out of service"

You can find general information about the "Out of service" mode in the Out of service (Page 77) chapter.

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit5` =0 and `Feature.Bit5` =1 is as shown below:

| Output Parameters | **`Feature.Bit5` =0** | **`Feature.Bit5` =1** |
|---|---|---|
| SP_ExtAct.ST | 16#80 | 16#60 |
| SP.ST | 16#80 | 16#60 |
| PV_Out.ST | 16#80 | 16#60 |
| PV_AH_Act.ST | 16#XX | 16#60 |
| PV_WH_Act.ST | 16#XX | 16#60 |
| PV_WL_Act.ST | 16#XX | 16#60 |
| PV_AL_Act.ST | 16#XX | 16#60 |
| PV_ToleHi.ST | 16#80 | 16#60 |
| PV_ToleLo.ST | 16#80 | 16#60 |
| ER.ST | 16#80 | 16#60 |

### See also

Program mode for controllers (Page 85)
PIDConS block diagram (Page 1091)
Description of PIDConS (Page 1069)
PIDConS functions (Page 1073)
PIDConS error handling (Page 1079)
PIDConS messaging (Page 1080)
PIDConS I/Os (Page 1082)

## 6.8.3 PIDConS functions

### Functions of PIDConS

The functions for this block are listed below.

### Generation of manipulated variables

The manipulated variable MV can be generated as follows:

| ManAct | MV_TrkOn | AdvCoAct AND NOT AdvCoModSP | MV = | Limit monitoring | State |
|---|---|---|---|---|---|
| 1 | - | - | Man | MV_HiLim<br>MV_LoLim | Manual mode, set by the operator |
| 0 | 1 | - | MV_Trk | MV_HiLim<br>MV_LoLim | Tracking with limitation |
| 0 | 0 | 0 | P_Part<br>+ I_Part<br>+ D_Part | MV_HiLim<br>MV_LoLim | Automatic mode (PID algorithm) |

If the controller is in "out of service" mode, the output parameter MV is set to the last valid value in manual mode or the neutral position manipulated variable depending on the Feature Bit (Neutral position manipulated variable takes effect at startup (Page 176)). Refer to the Out of service (Page 77) section for more on this.

### Tracking and limiting a manipulated variable

The block provides the standard function Tracking and limiting a manipulated variable (Page 222).

### Neutral position

The block provides the standard function Neutral position for motors, valves and controllers (Page 54).

### Group error

This block provides the standard function Outputting group errors (Page 130).

The following parameters are taken into consideration when forming the group error:

- CSF

### Outputting a signal for start readiness

This block provides the standard function Outputting a signal for start readiness (Page 59).

### "Actuator active" information

If the manipulated variable MV is greater than the minimum manual limit MV_LoLim.Value, this is recognized as actuator active. This status can be used to indicate a customized symbol in the process image, for example, and is saved in the status word (see Status word section in Description of PIDConS (Page 1069)).

### External/internal setpoint specification

The block provides the standard function Setpoint specification - internal/external (Page 136).

### Setpoint limitation

With this function, you can limit the setpoint to a range by means of the parameters `SP_HiLim` (high limit) and `SP_LoLim` (low limit). If the setpoint lies outside the range defined by you, it is limited to the valid range.

If the setpoint is at or above the limit `SP_HiLim`, this is displayed at the output `SP_HiAct = 1`.

If the setpoint is at or below the limit `SP_LoLim`, this is displayed at the output `SP_LoAct = 1`

### Tracking setpoint in manual mode

The block provides the standard function Tracking setpoint in manual mode (Page 222).

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Process value (`SimPV`)

### Limit monitoring of the process value

The block provides the standard function Limit monitoring of the process value (Page 93).

### Error signal generation and dead band

The block provides the standard function Error signal generation and dead band (Page 217).

### Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 240).

### PID algorithm

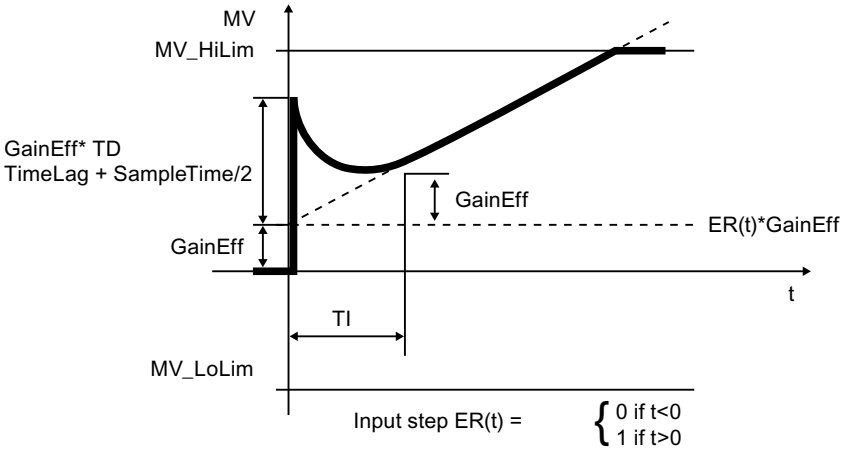
The manipulated variable is generated in automatic mode according to the following algorithm:

$MV = GainEff \cdot (1 + (1 / (TI \cdot s) + (TD \cdot s) / (1 + TD / DiffGain \cdot s)) \cdot ER$

Where:

s = Complex number

The following step response occurs:




**Note**

The formula describes a standard application where P, I and D components are activated and the P and D components are not in the feedback circuit (`PropSel = 1, TI <> 0, DiffToFbk = 0` and `PropFacSP = 1`).

The D component delay is derived from `TD` / `DiffGain`.

- The P component is displayed at the `P_Part` I/O and can be deactivated using `PropSel` = 0 .
- The I component is displayed at the `I_Part` I/O and can be deactivated using `TI` = 0 .
- The D component is displayed at the `D_Part` I/O and can be deactivated using `TD` = 0 .

## Anti-windup

The controller has an anti-windup function. The I component is frozen after the manipulated variable has reached limits (`MV_HiLim` or `MV_LoLim`).

## Forming the signal status for blocks

The block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

- Signal status for the process value `PV_Out`:
  The signal status of the output parameter `PV_Out` always corresponds to the signal status of input parameter `PV` or, if the block is in simulation mode, 16#60.
- Signal status for the setpoint value `SP`:
  The signal status of the `SP` output parameter is always equivalent to the signal status of the input parameter `SP_Ext` or `SP_Int`, depending on how the setpoint is specified. If the internal setpoint `SP_Int` is used, the signal status is always outputted as `16#80`.
- Signal status of the error signal `ER`:
  The signal status of output parameter `ER` is obtained from the worst signal status of the two output parameters `PV_Out` and `SP`, and is outputted.
  The signal status `16#60` (external simulation) is suppressed because the block acts as a sink with external simulation.

- Signal status for the manipulated variable MV:
  The signal status of output parameter MV is formed in "Automatic mode" from the error signal ER and output. In "manual mode", the signal status is output as good. The signal status 16#60 (external simulation) is suppressed because the block acts as a sink with external simulation. In "manual mode", the signal status is output as good.
- Worst signal status:
  The worst signal status ST_Worst for the block is formed from SP, PV_Out, MV_ChnST and Rbk .

### Configurable reactions using the Feature parameter

You can find an overview of all reactions provided by the Feature parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 3 | Disable limit act output when limit value message is disabled (Page 209) |
| 4 | Not used. |
| 5 | Changes signal status of outputs in OOS (Page 208) |
| 6-14 | Not used. |
| 15 | Neutral position manipulated variable takes effect with "out of service" operating mode (Page 177) |
| 16 | Neutral position manipulated variable takes effect at startup (Page 176) |
| 17 - 22 | Not used. |
| 23 | SP following PV in open loop has no priority over SP_Ext and SP limits (Page 193) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 27 - 30 | Not used. |
| 31 | Limits disabled in faceplate (Page 206) |

### Operator permissions

The block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the OS_Perm parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" AutModOp |
| 1 | 1 = Operator can switch to "manual mode" ManModOp |
| 2 | 1 = Operator can switch to "Out of service" mode OosOp |
| 3 | 1 = Operator can switch the setpoint to "external" SP_ExtOp |
| 4 | 1 = Operator can switch the setpoint to "internal" SP_IntOp |
| 5 | 1 = Operator can change the internal setpoint SP_Int |

| Bit | Function |
|---|---|
| 6 | 1 = Operator can change the manual parameter `Man` |
| 7 | 1 = Operator can change the high operation limit of the setpoint `SP_HiLim` |
| 8 | 1 = Operator can change the low operation limit of the setpoint `SP_LoLim` |
| 9 | 1 = Operator can change the operation high limit of the manipulated variable `ManHiLim` |
| 10 | 1 = Operator can change the operation low limit of the manipulated variable `ManLoLim` |
| 11 | 1 = Operator can permit the PID optimization function `OptimEn` |
| 12 | 1 = Operator can enable the track setpoint in "manual mode" function `SP_TrkPV` |
| 13 | 1 = Operator can enable the bumpless switchover from external to internal `SP_TrkExt` |
| 14 | 1 = Operator can change the gain parameter `Gain` |
| 15 | 1 = Operator can change the integral time parameter `TI` |
| 16 | 1 = Operator can change the derivative time parameter `TD` |
| 17 | 1 = Operator can change the derivative gain parameter `DiffGain` |
| 18 | 1 = Operator can change the dead band parameter `DeadBand` |
| 19 | 1 = Operator can change the limit (process value) `PV_AH_Lim` for the high alarm |
| 20 | 1 = Operator can change the limit (process value) `PV_WH_Lim` for the high warning |
| 21 | 1 = Operator can change the hysteresis (process value) `PV_Hyst` |
| 22 | 1 = Operator can change the limit (process value) `PV_WL_Lim` for the low warning |
| 23 | 1 = Operator can change the limit (process value) `PV_AL_Lim` for the low alarm |
| 24 | 1 = Operator can activate the Simulation function `SimOn` |
| 25 | 1 = Operator can activate the Release for maintenance function `MS_RelOp` |
| 26 | 1 = Operator can change the simulation value `SimPV` |
| 27 | 1 = Operator can activate / deactivate messages via `PV_AH_MsgEn` |
| 28 | 1 = Operator can activate / deactivate messages via `PV_WH_MsgEn` |
| 29 | 1 = Operator can activate / deactivate messages via `PV_WL_MsgEn` |
| 30 | 1 = Operator can activate / deactivate messages via `PV_AL_MsgEn` |
| 31 | Not used |

The block has the following permissions for the `OS1Perm` parameter:

| Bit | Function |
|---|---|
| 0 - 15 | reserved |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

## Release for maintenance

The block provides the standard function Release for maintenance (Page 70)

### Generating instance-specific messages

The block provides the standard function Generating instance-specific messages (Page 230) without the time stamp function in the I/O.

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Specifying the display area for process and setpoint values as well as operations

This block provides the standard function Display and operator input area for process values and setpoints (Page 233).

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Labeling of buttons and text (Page 235)

EventTs functions (Page 2046)

PIDConS modes (Page 1072)

PIDConS error handling (Page 1079)

PIDConS messaging (Page 1080)

PIDConS I/Os (Page 1082)

PIDConS block diagram (Page 1091)

## 6.8.4 PIDConS error handling

### Error handling of PIDConS

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Control system fault (CSF)

### Overview of error numbers

The `ErrorNum` output parameter can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. |
| 30 | The value of `PV` can no longer be displayed in the REAL number field. |
| 31 | The value of `SP_Ext` can no longer be displayed in the REAL number field. |
| 33 | The value of `MV_Trk` can no longer be displayed in the REAL number field. |
| 35 | The value of `Rbk` can no longer be displayed in the REAL number field. |
| 36 | The value of `MV` can no longer be displayed in the REAL number field. |
| 51 | `AutModLi` = 1 and `ManModLi` = 1<br>`SP_LiOp` = 1 and `SP_IntLi` = 1 and `SP_ExtLi` = 1 |
| 60 | \|`TI`\| < `SampleTime` / 2 |
| 61 | \|`TD`\| < `SampleTime` |
| 62 | `DiffGain` < 1 or `DiffGain` > 10 |
| 63 | `TD` / `DiffGain` < `SampleTime` / 2 |

### Control system fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the section Error handling (Page 128).

### See also

Description of PIDConS (Page 1069)

PIDConS modes (Page 1072)

PIDConS functions (Page 1073)

PIDConS messaging (Page 1080)

PIDConS I/Os (Page 1082)

PIDConS block diagram (Page 1091)

## 6.8.5 PIDConS messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Process messages
- Instance-specific messages

### Process control fault

The following control system fault messages can be output:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 5 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If it changes to CSF = 1, a process control fault is triggered (MsgEvId1, SIG 5).

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 1 | Alarm - high | $$BlockComment$$<br>PV - high alarm limit violated |
| | SIG 2 | Warning - high | $$BlockComment$$<br>PV - high warning limit violated |
| | SIG 3 | Warning - low | $$BlockComment$$<br>PV - low warning limit violated |
| | SIG 4 | Alarm - low | $$BlockComment$$<br>PV - low alarm limit violated |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Instance-specific messages

You have the option of using one or two instance-specific messages for this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 6 | AS process control message - fault | $$BlockComment$$<br>External message 1<br>Status 16#@8%x@ |
| | SIG 7 | AS process control message - fault | $$BlockComment$$<br>External message 2<br>Status 16#@9%x@ |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance `MsgEvId1`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | Process value `PV_Out` |
| 5 | Control deviation `ER` |
| 6 | Signal status `ExtMsg1` |
| 7 | Signal status `ExtMsg2` |
| 8 | `ExtVa108` |
| 9 | `ExtVa109` |
| 10 | Not allocated |

The associated values 8 ... 9 are allocated to the parameters `ExtVa108` ... `ExtVa109` and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of PIDConS (Page 1069)

PIDConS modes (Page 1072)

PIDConS functions (Page 1073)

PIDConS error handling (Page 1079)

PIDConS I/Os (Page 1082)

PIDConS block diagram (Page 1091)

## 6.8.6 PIDConS I/Os

I/Os of PIDConS

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `AutModLi*` | 1 = "Automatic mode" via interconnection or SFC (controlled by `ModLiOp` = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `AutModOp*` | 1 = "Automatic mode" via operator (controlled by `ModLiOp` = 0) | BOOL | 0 |
| `BatchEn` | 1 = Enable allocation for batch control | BOOL | 0 |
| `BatchID` | Batch ID | DWORD | 16#00000000 |
| `BatchName` | Batch name | S7-String | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| BlkTypeInfo* | Block Type Version and FB Number of block-type<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| CPI_In | Input for control performance index, which is calculated by the assigned ConPerMon block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#78 |
| CSF | 1 = External error (control system fault)<br>Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DeadBand | Width of dead band | REAL | 0.0 |
| DiffGain | Gain of differentiator [1..10] DiffGain = TD / (delay time of D component) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 5.0<br>• 16#80 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| ExtMsg1 | 1 = Binary input for freely selectable message 1 is used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | 1 = Binary input for freely selectable message 2 is used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa108 | Associated value 8 for messages (MsgEvID1) | ANY | |
| ExtVa109 | Associated value 9 for messages (MsgEvID1) | ANY | |
| Feature | I/O for additional PIDConS functions (Page 1073) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| Gain | Proportional gain<br>Gain.ST = 16#FF: Enabled in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#FF |
| IntHoldNeg | 1 = Integrator cannot run in negative direction | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| IntHoldPos | 1 = Integrator cannot run in positive direction | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Man* | Manual specification for the manipulated variable | REAL | 0.0 |
| ManModLi* | 1 = "Manual mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManModOp* | 1 = "Manual mode" via OS operator (controlled by ModLiOp = 0) | BOOL | 1 |
| ModLiOp | Operating mode switchover between:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_RelOp* | 1 = Release for maintenance by OS operator | BOOL | 0 |
| MsgEvID1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_ChnST | Signal status of output channel for MV<br>Should be connected to an output channel block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| MV_HiLim | Limit (high) for manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| MV_LoLim | Limit (low) for manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_Offset | Manipulated variable for ER=0, operating point for controller with deactivated I component | REAL | 0.0 |
| MV_OpScale | OS display range for manipulated variable MV | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| MV_SafePos | Manipulated variable neutral position | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_Trk* | Tracking value for the manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_TrkOn | 1 = Tracking of manipulated variable MV | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MV_Unit | Unit of measure for manipulated variable | INT | 1342 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OptimEn* | 1 = Enable optimization of PID parameters by PID tuner | BOOL | 0 |
| OptimOcc* | 1 = Optimization running | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permission (Page 1073) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| OS1Perm | I/O for operator permissions (Page 1073) | STRUCT<br>• Bit 0: BOOL<br>• Bit 15: BOOL | -<br>• 1<br>• 1 |
| PropSel | 1 = Activate P component | BOOL | 1 |
| PV* | Process value (controlled variable) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_A_DC* | Delay time for incoming PV alarms [s] | REAL | 0.0 |
| PV_AH_En | 1 = Enable PV alarm limit (high) | BOOL | 1 |
| PV_AH_Lim | Limit PV alarm (high) | REAL | 95.0 |
| PV_AH_MsgEn | 1 = Enable PV alarm (high) message | BOOL | 1 |
| PV_AL_En | 1 = Enable PV alarm limit (low) | BOOL | 1 |
| PV_AL_Lim | PV alarm limit (low) | REAL | 5.0 |
| PV_AL_MsgEn | 1 = Enable PV alarm (low) message | BOOL | 1 |
| PV_Hyst | Hysteresis for PV alarm, warning and tolerance limits | REAL | 1.0 |
| PV_OpScale | Limit for scale in PV bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| PV_Unit | Unit of measure for process value | INT | 1001 |
| PV_W_DC* | Delay time for incoming PV warnings [s] | REAL | 0.0 |
| PV_WH_En | 1 = Enable PV warning limit (high) | BOOL | 1 |
| PV_WH_Lim | Limit PV warning (high) | REAL | 90.0 |
| PV_WH_MsgEn | 1 = Enable PV warning (high) message | BOOL | 1 |
| PV_WL_En | 1 = Enable PV warning limit (low) | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_WL_Lim | Limit PV warning (low) | REAL | 10.0 |
| PV_WL_MsgEn | 1 = Enable PV warning (low) message | BOOL | 1 |
| Rbk* | Position feedback for display on OS | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SafePos | 1 = Neutral position (Page 54) for controller manipulated variable is MV_HiLim<br>0 = Neutral position for controller manipulated variable is MV_LoLim | BOOL | 0 |
| SafePos2 | Neutral position for control valve:<br>0 = SafePos is valid<br>1 = Neutral position is MV_SafePos<br>2 = Neutral position is last manipulated variable (stop) | INT | 0 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | 1 = Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SimPV* | Process value used for SimOn = 1 | REAL | 0.0 |
| SP_Ext* | external setpoint - (to interconnection) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_ExtLi* | 1 = Select external setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtOp* | 1 = Select external setpoint (via operator) | BOOL | 0 |
| SP_HiLim | Limit (high) of internal SP | STRUCT<br>• Value: BOOL<br>• ST: BYTE | <br>• 100.0<br>• 16#80 |
| SP_Int* | Internal setpoint for operation | REAL | 0.0 |
| SP_IntLi* | 1 = Select internal setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_IntOp* | 1 = Select internal setpoint (via operator) | BOOL | 0 |
| SP_LiOp | Select setpoint source (internal/external):<br>1 = Via interconnection<br>0 = Via operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP_LoLim | Limit (low) of internal SP | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_TrkPV | 1 = Setpoint follows PV in "manual mode" and with tracking | BOOL | 0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| TD | Derivative time [s]<br>TD.ST = 16#FF: Enabled in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| TI | Integral time [s]<br>TI.ST = 16#FF: Enabled in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#FF |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutAct | 1 = "Automatic mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| D_Part | D component of PID algorithm | REAL | 0.0 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ER | Error signal | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see PIDConS error handling (Page 1079) | INT | -1 |
| GrpErr | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| I_Part | I component of PID algorithm | REAL | 0.0 |
| LoopClosed | 1 = Control loop closed<br>0 = Control loop open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ManAct | 1 = "Manual mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV | Manipulated variable | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_HiAct | 1 = Limit (high) of manipulated variable violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_LoAct | 1 = Limit (low) of manipulated variable violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_UnitOut | Unit of measure for manipulated variable, for interconnecting to the MV_Unit input parameter of the ConPerMon block | INT | 0 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1_Perm | WORD | 16#FFFF |
| OS1PermLog | Display of OS1_Perm with settings changed by the block algorithm | WORD | 16#FFFF |
| P_Part | P component of PID algorithm | REAL | 0.0 |
| PV_AH_Act | 1 = PV alarm (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_AL_Act | 1 = PV alarm (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_Out | Output for process value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_OpScaleOut | PV_OpScale as output | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| PV_ToleHi | Limit (high) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_ToleLo | Limit (low) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_WH_Act | 1 = PV warning (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_WL_Act | 1 = PV warning (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_UnitOut | Unit of measure for process value, for interconnecting to the PV_Unit input parameter of the ConPerMon block | INT | 0 |
| RdyToStart | 1 = Active start readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP | Setpoint used by controller | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_ExtAct | 1 = External setpoint is active<br>0 = Internal setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_HiOut | Limit (high) for SP corresponds to input parameter SP_HiLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP_LoOut | Limit (low) for SP corresponds to input parameter SP_LoLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 1069) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1069) | DWORD | 16#00000000 |
| SumMsgAct | 1 = Active hardware interrupt | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

## See also

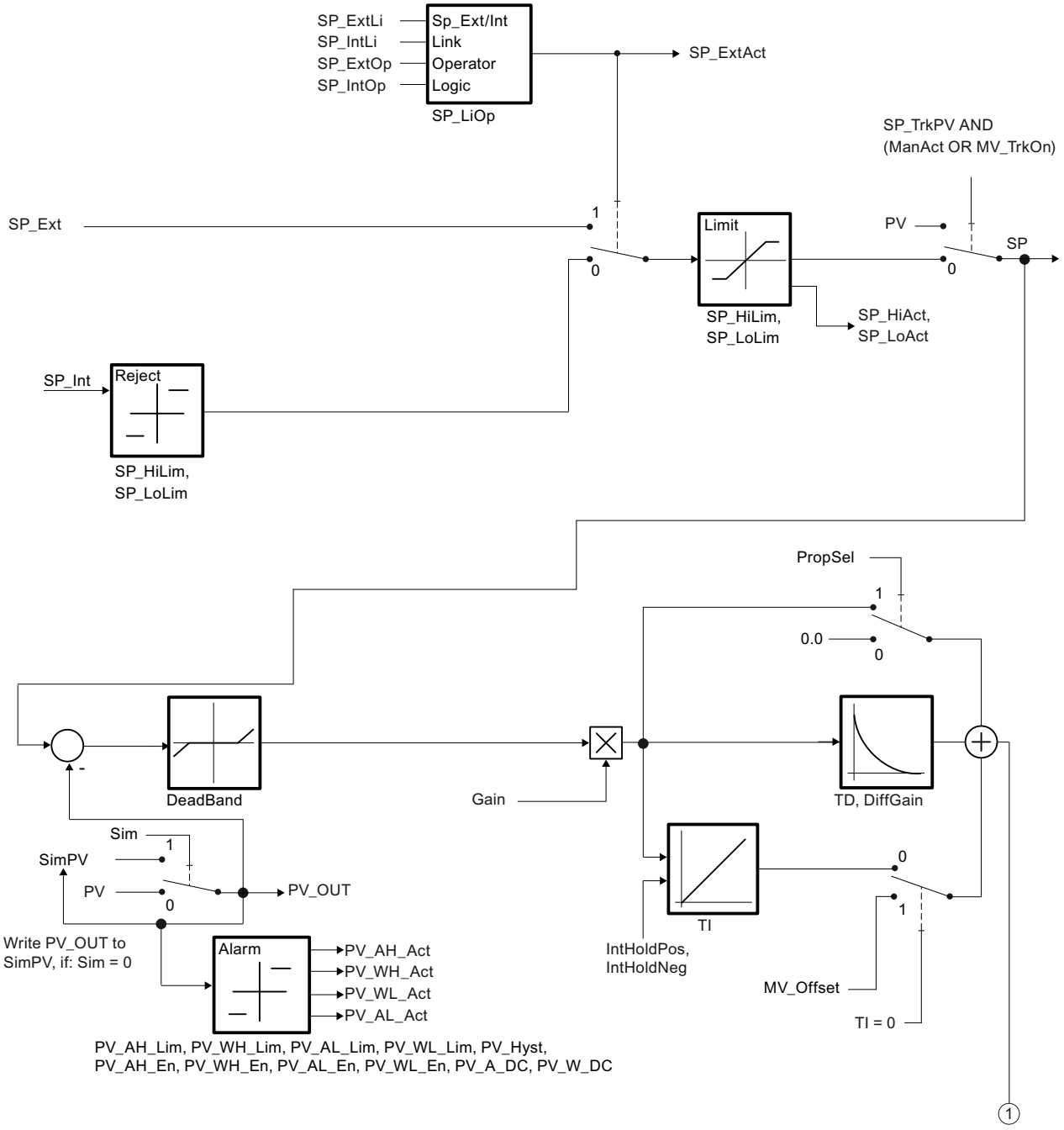
PIDConS modes (Page 1072)

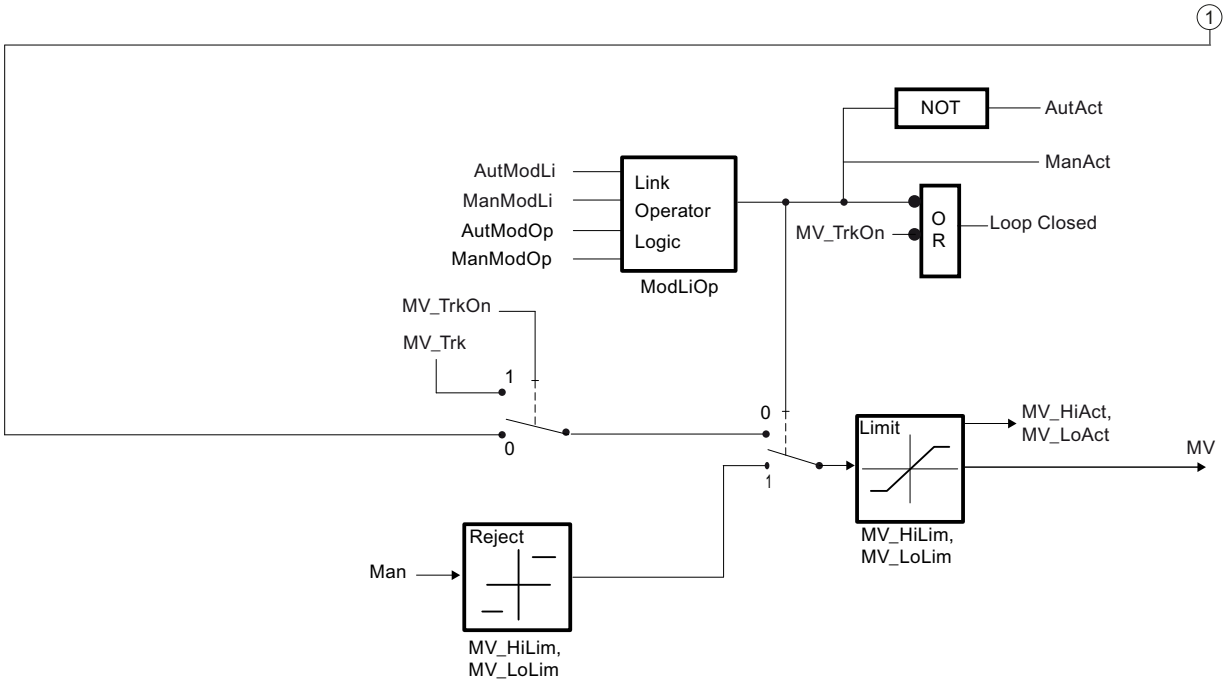
PIDConS messaging (Page 1080)

PIDConS block diagram (Page 1091)

## 6.8.7 PIDConS block diagram

### PIDConS block diagram






**See also**

PIDConS I/Os (Page 1082)

PIDConS messaging (Page 1080)

PIDConS error handling (Page 1079)

PIDConS functions (Page 1073)

PIDConS modes (Page 1072)

Description of PIDConS (Page 1069)

## 6.8.8 Operator control and monitoring

### 6.8.8.1 PIDConS views

#### Views of the PIDConS block

The block PIDConS provides the following views:

- PIDConL, PIDConS and PIDConR standard views (Page 1061)
- Alarm view (Page 336)
- Limit view of PID controllers (Page 325)
- Trend view (Page 341)
- Parameter view of PID controllers (Page 315)
- PIDConL, PIDConS and PIDConR previews (Page 1066)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icons for PID and FM controller (Page 269)

Refer to the Structure of the faceplate (Page 279) and Block icon structure (Page 260) chapters for general information about the faceplate and block icon.

#### See also

Ramp view (Page 334)

# 6.9 PIDConR - Continuous PID controller with external reset

## 6.9.1 Description of PIDConR

### Object name (type + number) and family

Type + number: FB 1875

Family: Control

### Area of application for PIDConR

The block is used for the following applications:

- Fixed setpoint control
- Cascade control
- Ratio control
- Split-range control
- Smith predictor closed-loop control
- Override control (override)

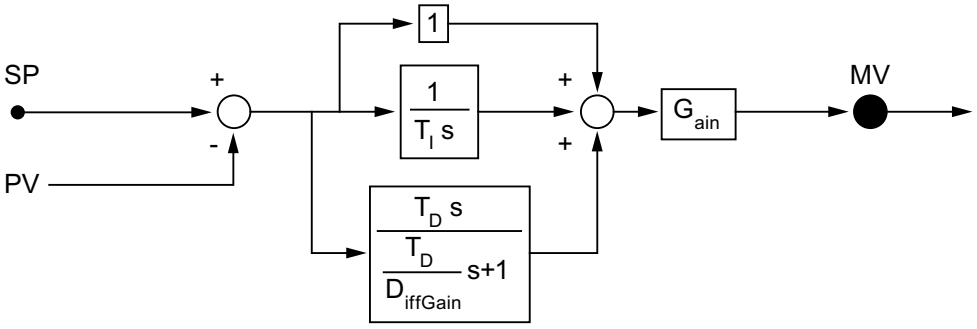
Unlike PIDConL, the PIDConR permits an external reset and satisfies the special requirements of the US market.

### How it works

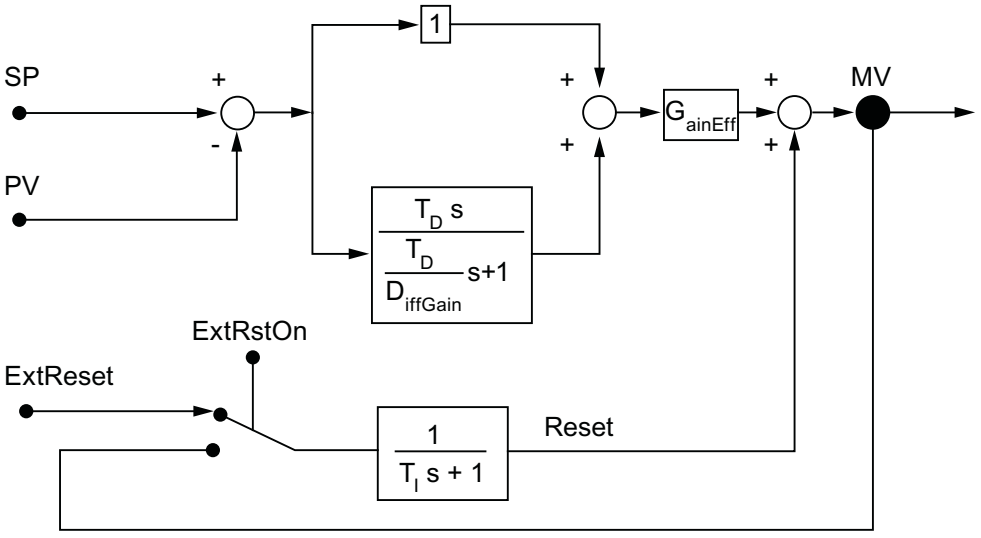
The block is a PID controller with continuous output signal (manipulated variable). It is used to activate a final controlling element with continuous action input.

The block functions following the PID algorithm with a delayed D component and an integrator with double precision. Its particular feature is that it is an incremental control algorithm with a serial-interactive structure. Incremental means that the current manipulated variable is calculated from the old manipulated variable of the last sampling step. In place of the old manipulated variable, an output point for the manipulated variable calculation (external reset) can be specified externally by interconnection. The difference between the parallel controller structure of the PIDConL and the serial-interactive structure of the PIDConR is shown in the next two diagrams.

**PIDConL**




**PIDConR**




The block is suitable for controlling sluggish control loops, for example, for temperatures and filling levels, and high-speed control loops, for example, for flow rates and speed. For a given CPU, a compromise has to be made between the number of controllers and the frequency with which the individual controllers have to be processed. The faster the modulated control loops are, i.e. the more frequently the manipulated variables have to be calculated per time unit, the lower the number of controllers that can be installed.

## Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

For the control loop monitoring to work as planned in the trend view of the controller faceplates, the

```
S7_xarchive:='Value, shortterm;'
```

attributes in the process tag types for control loops at the controller function block must be set for the following tags:

- Input parameters:
  - CPI_In
- Output parameters
  - MV
  - MV_HiAct
  - MV_LoAct
  - LoopClosed
  - SP
  - PV_Out
  - PV_ToleHi
  - PV_ToleLo

For the PIDConR block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Cascade control with PIDConR (CascadeR) (Page 2843)
- Override control with PIDConR (OverrideR) (Page 2848)
- PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 2823)
- Ratio control with PIDConR (RatioR) (Page 2839)

**Note**

The meaning of the controller parameters of both structures is different for all controller parameter settings with a D component. If you want to transfer parameter values from one structure to another, they have to be converted according to the formula in the PIDConR function .

### Startup characteristics

Use the Feature Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at RunUpCyc.

The following output parameters are written with the corresponding input parameters:

- PV_HysOut
- PV_AH_Out
- PV_WH_Out
- PV_TH_Out
- PV_AL_Out

- PV_WL_Out
- PV_TL_Out

### Status word allocation for Status1 parameter

You can find a description for each parameter in section PIDConR I/Os (Page 1121).

| Status bit | Parameter |
|---|---|
| 0 | Occupied |
| 1 | BatchEn |
| 2 | SimOn |
| 3 | OosAct.Value |
| 4 | OosLi.Value |
| 5 | AutAct.Value |
| 6 | Invalid signal status |
| 7 | ManAct.Value |
| 8 | SP_ExtAct.Value |
| 9 | MV_ForOn.Value |
| 10 | MV_TrkOn.Value AND NOT (ManAct.Value OR OosAct.Value OR MV_ForOn.Value) |
| 11 | MV.Value > ManLoLim |
| 12 | SimLiOp.Value |
| 13 | "Interlock" button is enabled OR Intlock.ST ≠ 16#FF |
| 14 | 0 = Open padlock in the block icon<br>1 = Closed padlock in block icon |
| 15 | SP_LoadOn.Value |
| 16 | SP_LoadOn.Value AND NOT (MV_TrkOn.Value OR OosAct.Value OR MV_ForOn.Value) |
| 17 | Feature.Bit19 |
| 18 | Feature.Bit21 |
| 19 | AdvCoAct |
| 20 | 1 = Input parameter Rbk is not interconnected (RbkOut.ST = 16#FF) |
| 21 | NegGain |
| 22 | 1 = SP ramp active |
| 23 | OptimEn |
| 24 | OptimOcc |
| 25 | Not used |
| 26 | Display of BypassAct.Value in faceplate (display and operator controls) and block icon |
| 27 | AdvCoModSP |
| 28 | SimOn AND ManAct |
| 29 | BypProt |
| 30 | Dead band is temporarily disabled |
| 31 | Not used |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `MsgLock` |
| 1 | `PV_AH_Act.Value` |
| 2 | `PV_WH_Act.Value` |
| 3 | `PV_TH_Act.Value` |
| 4 | `PV_TL_Act.Value` |
| 5 | `PV_WL_Act.Value` |
| 6 | `PV_AL_Act.Value` |
| 7 | `PV_AH_En` OR `Feature.Bit31` |
| 8 | `PV_WH_En` OR `Feature.Bit31` |
| 9 | `PV_TH_En` OR `Feature.Bit31` |
| 10 | `PV_TL_En` OR `Feature.Bit31` |
| 11 | `PV_WL_En` OR `Feature.Bit31` |
| 12 | `PV_AL_En` OR `Feature.Bit31` |
| 13 | `PV_AH_MsgEn` |
| 14 | `PV_WH_MsgEn` |
| 15 | `PV_TH_MsgEn` |
| 16 | `PV_TL_MsgEn` |
| 17 | `PV_WL_MsgEn` |
| 18 | `PV_AL_MsgEn` |
| 19 | `ER_AH_Act.Value` |
| 20 | `ER_AL_Act.Value` |
| 21 | `ER_AH_En` OR `Feature.Bit31` |
| 22 | `ER_AL_En` OR `Feature.Bit31` |
| 23 | `ER_AH_MsgEn` |
| 24 | `ER_AL_MsgEn` |
| 25 | `RbkWH_Act.Value` |
| 26 | `RbkWL_Act.Value` |
| 27 | `RbkWH_En` OR `Feature.Bit31` |
| 28 | `RbkWL_En` OR `Feature.Bit31` |
| 29 | `RbkWH_MsgEn` |
| 30 | `RbkWL_MsgEn` |
| 31 | `MS_RelOp` |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 | Effective signal 1 of the message block connected via `EventTsIn` |
| 1 | Effective signal 2 of the message block connected via `EventTsIn` |
| 2 | Effective signal 3 of the message block connected via `EventTsIn` |
| 3 | Effective signal 4 of the message block connected via `EventTsIn` |
| 4 | Effective signal 5 of the message block connected via `EventTsIn` |

| Status bit | Parameter |
|---|---|
| 5 | Effective signal 6 of the message block connected via `EventTsIn` |
| 6 | Effective signal 7 of the message block connected via `EventTsIn` |
| 7 | Effective signal 8 of the message block connected via `EventTsIn` |
| 8 | Delay of the `PV_AH_Lim` message |
| 9 | Delay of the `PV_WH_Lim` message |
| 10 | Delay of the `PV_TH_Lim` message |
| 11 | Delay of the `PV_TL_Lim` message |
| 12 | Delay of the `PV_WL_Lim` message |
| 13 | Delay of the `PV_AL_Lim` message |
| 14 | Delay of the `ER_AH_Lim` message |
| 15 | Delay of the `ER_AL_Lim` message |
| 16 | Delay of the `RbkWH_Lim` message |
| 17 | `BypassAct.Value` |
| 18 | Collection of message delays |
| 19 | Delay of the `RbkWL_Lim` message |
| 20 | 1= Suppressed messages |
| 21 - 23 | Not used |
| 24 | `Feature2.Bit2`: Separate bypass signal |
| 25 | Hidden bypass signal in Intlock |
| 26 | `LockAct.Value` |
| 27 | `SP_UpRaAct`, `SP_DnRaAct` limits enabled for gradient mode (`SP_RateOn = 1`) |
| 28 | `GrpErr.Value` |
| 29 | `RdyToStart.Value` |
| 30 | 1 = Input parameter `MV_ChnST` is interconnected |
| 31 | Internal error |

## Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 - 7 | Effective signal 9..16 of the message block connected via `EventTsIn` |
| 8 - 28 | Not used |
| 29 | 1 = Input parameter `SP_LiRmpTime` is not interconnected (`SP_LiRmpTime.ST = 16#FF`) |
| 30 | Auxiliary value 1 visible |
| 31 | Auxiliary value 2 visible |

## See also

PIDConR block diagram (Page 1140)

PIDConR messaging (Page 1118)

PIDConR error handling (Page 1117)

PIDConR functions (Page 1103)

PIDConR modes (Page 1100)

## 6.9.2 PIDConR modes

### Operating modes of PIDConR

The block can be operated using the following modes:

- Automatic mode (Page 79)
- Manual mode (Page 79)
- Program mode for controllers (Page 85)
- Out of service (Page 77)

In "manual mode", the control settings for the device are made manually by the operator. The operator decides how to change the block's manipulated variable (output signal).

In "automatic mode", the controller's manipulated variable is calculated automatically by the block algorithm.

### Changing between operating modes

The switchover between manual and automatic modes takes place as shown in the following schematic:

**Switchover by using faceplates:** The switchover between operating modes is carried out in the standard view of the faceplate. In the function block, the parameters `ManModOp` for "manual mode" and `AutModOp` for "automatic mode" are used.

**Switchover via interconnection (CFC or SFC instance):** The switchover between the operating modes is carried out by means of interconnection on the function block.

**Note**

You can access the variable parameters `AutModOp` and `ManModOp` from a normal SFC (in contrast to the instance of an SFC type). The SFC can thus change the operating mode without revoking the access rights of the operator.

## Points to note for this block

With PIDConR the switchover between operating modes via interconnectable input parameters follows a different logic than that used for other controller blocks. This logic is oriented towards the special requirements of the US market. The basic approach is to issue the controller with certain commands by interconnection using an individual input parameter. The commands issued by interconnection take priority over the entries in the faceplate, i.e. if such a command input is made, the associated operator controls are locked in the faceplate:

- If the interconnectable `AutExtSet` = 1 input parameter is set, the controller goes into "automatic mode" with an external setpoint. This is also known as the "cascade" operating mode.
- If the interconnectable `AutIntSet` = 1 input parameter is set, the controller goes into "automatic" mode with internal setpoint. `AutIntSet` has higher priority than`AutExtSet`. For more information about the internal and external setpoint, refer to section PIDConR functions (Page 1103).
- If the interconnectable input parameter `ManSet` = 1 is set, the controller goes into "manual" mode. This command has higher priority than `AutIntSet` and `AutExtSet`.
- If one of the interconnectable input parameters `MV_Close` or `MV_Open` is set, the controller also goes into "manual" operating mode and performs the corresponding command.

The `ModLiOp`, `ManModLi` and `AutModLi` input parameters are not therefore present in PIDConR.

## Switchover from automatic mode to manual mode

When changing over from "automatic mode" to "manual mode", the last valid control settings (**M**anipiulated **V**alue `MV`) for the controller set in "automatic mode" remain valid until you change the control settings manually.

## Switchover from manual mode to automatic mode

The switchover from manual to automatic mode can take place with or without the internal setpoint tracking the process value. You specify this behavior on the `SP_TrkPV` I/O, which can also be operated from the faceplate in the parameter view (Option "`SP` : = `PV` in Manual").

### Switchover with tracked internal setpoint

(`SP_TrkPV` = 1) means that in "manual" mode the setpoint (`SP`) tracks the process variable (`PV`). This results in bumpless switchover, which means the manipulated variable remains constant after switching back to "automatic mode" until the setpoint (`SP`) is changed or the process value (`PV`) changes.

### Switchover without tracked internal setpoint

(`SP_TrkPV` = 0) means that the setpoint does not track the actual value in manual mode and that the block immediately recalculates the manipulated variable based on the setpoint and process value (`PV`) when the mode is changed. PIDConR only offers **switchover without P step**: During switchover, the I action (reset) of the controller is set in such a way that the switchover is carried out without a P step (virtually bumpless referring to the manipulated variable). In case of a large control deviation, when switching from the manual mode to the automatic mode, it may happen that the integral action is set far outside the manipulated variable limits. See

`Feature` bit With accelerated return of the integral action from the manipulated variable limit (Page 193).

### Reaction of signals when operating mode is changed

Using the `Feature` Bit Resetting the commands for changing the mode (Page 172), you can specify whether the block automatically resets the signal for changing the operating mode.

### "Program mode for controllers"

You can find general information about the "Program mode for controller" in the section Program mode for controllers (Page 85).

In case of a large control deviation, when switching from the program mode with manipulated variable specification to the automatic mode, it may happen that the integral action is set far outside the manipulated variable limits. See `Feature` bit With accelerated return of the integral action from the manipulated variable limit (Page 193).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit5` =0 and `Feature.Bit5` =1 is as shown below:

| Output Parameters | **`Feature.Bit5` =0** | **`Feature.Bit5` =1** |
|---|---|---|
| RbkOut.ST | 16#80 | 16#60 |
| RbkWH_Act.ST | 16#XX | 16#60 |
| RbkWL_Act.ST | 16#XX | 16#60 |
| SP_ExtAct.ST | 16#80 | 16#60 |
| SP.ST | 16#80 | 16#60 |
| SP_ExtOut.ST | 16#80 | 16#60 |
| PV_Out.ST | 16#80 | 16#60 |
| PV_AH_Act.ST | 16#XX | 16#60 |
| PV_WH_Act.ST | 16#XX | 16#60 |
| PV_TH_Act.ST | 16#XX | 16#60 |
| PV_TL_Act.ST | 16#XX | 16#60 |
| PV_WL_Act.ST | 16#XX | 16#60 |
| PV_AL_Act.ST | 16#XX | 16#60 |
| PV_ToleHi.ST | 16#80 | 16#60 |
| PV_ToleLo.ST | 16#80 | 16#60 |
| ER.ST | 16#80 | 16#60 |
| ER_AH_Act.ST | 16#XX | 16#60 |
| ER_AL_Act.ST | 16#XX | 16#60 |

**Note**

The status of `RbkOut` will be 16#FF without Rbk. when `Feature.Bit5` = 0.

### See also

PIDConR block diagram (Page 1140)

PIDConR I/Os (Page 1121)

PIDConR messaging (Page 1118)

PIDConR error handling (Page 1117)

Description of PIDConR (Page 1094)

Disabling bumpless switchover to automatic mode for controllers (Page 183)

## 6.9.3 PIDConR functions

### Functions of PIDConR

The functions for this block are listed below.

### Generation of manipulatedvariables

The manipulated variable `MV` can be generated as follows:

| MV_ForOn | MV_BumpOn | MV_Close | MV_Open | Man-Act | MV_TrkOn | Adv-CoAct AND NOT AdvCo-ModSP | MV = | Limit monitoring | State |
|---|---|---|---|---|---|---|---|---|---|
| 1 | - | - | - | - | - | - | `MV_Forced` | none | Forced tracking through constraint without limitation |
| 0 | 1 | - | - | - | - | - | `MV_Bump` | Manual mode:<br>`ManHiLim`<br>`ManLoLim`<br>Automatic mode:<br>`MV_HiLim`<br>`MV_LoLim` | Sudden manipulation of the manipulated variable |
| 0 | 0 | 1 | - | - | - | - | `ManLoLim` | `ManHiLim`<br>`ManLoLim` | Close by interconnection |

| MV_ForOn | MV_BumpOn | MV_Close | MV_Open | Man-Act | MV_TrkOn | Adv-CoAct AND NOT AdvCo-ModSP | MV = | Limit monitoring | State |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | - | - | - | ManHiLim | ManHiLim<br>ManLoLim | Open by interconnection |
| 0 | 0 | 0 | 0 | 1 | - | - | Man | ManHiLim<br>ManLoLim | Manual mode, set by the OS operator or via the ManSet = 1 command |
| 0 | 0 | 0 | 0 | 0 | 1 | - | MV_Trk | MV_HiLim<br>MV_LoLim | Tracking with limitation |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | AdvCoMV | MV_HiLim<br>MV_LoLim | Higher-level program mode |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | PID.OUT + FFwd | MV_HiLim<br>MV_LoLim | Automatic mode (PID algorithm) |

If the controller is in "out of service" mode, the output parameter MV is set to the last valid value in manual mode or the neutral position manipulated variable depending on the Feature Bit Neutral position manipulated variable takes effect at startup (Page 176). Refer to the Out of service (Page 77) section for more on this.

The PIDConR offers the following special ways of influencing the generation of manipulated variables via interconnectable input parameters. The commands issued by interconnection take priority over the entries in the faceplate, i.e. if such a command input is made, the associated operator controls are locked in the faceplate.

**MV_BumpOn** is used for sudden manipulation of the manipulated variable and has a similar effect to MV_ForOn, the difference being that the appropriate limits are applied. If MV_BumpOn = 1 is set in "automatic mode", the MV_Bump manipulated variable is written in a limited manner between MV_HiLim and MV_LoLim to output MV. If MV_BumpOn = 1 is set in "manual mode", the MV_Bump manipulated variable is written in a limited manner between ManHiLim and ManLoLim to output MV. If MV_BumpOn is reset to 0, the controller returns to its previous mode.

**MV_Close** is used to close the adjustment valve. MV_Close = 1 switches the controller to "manual mode" with manipulated variable MV = ManLoLim. If MV_Close is reset to 0, the controller remains in "manual mode".

**MV_Open** is used to open the adjustment valve. MV_Open = 1 switches the controller to "manual mode" with manipulated variable MV = ManHiLim. If MV_Open is reset to 0, the controller remains in "manual mode".

These commands have higher priority than "automatic mode", but lower priority than forced tracking by MV_ForOn.

For meshed controller structures, such as a cascade control and closed-loop control with the PIDConR block, the ExtReset input parameter is used with ExtRstOn = 1 rather than MV_Trk with MV_TrkOn = 1 (also refer to the process tag types Cascade control with PIDConR (CascadeR) (Page 2843) and Override control with PIDConR (OverrideR) (Page 2848)).

### Displaying additional information relating to the manipulated variable on the output

The manual manipulated variable limitations ManLoLim and ManHiLim are copied to the ManLoOut and ManHiOut output parameters so that they can be further interconnected to the secondary controller as setpoint limits SP_ExtLoLim and SP_ExtHiLim. If you want to use the same limit pairs for "manual" and "automatic" mode, you can interconnect output parameters ManLoOut and ManHiOut to input parameters MV_LoLim and MV_HiLim of the same block and thereby control the limits for manipulated variable limitation in the faceplate. In most cases, a valve's complete control range can be passed through in "manual" mode. You can then further interconnect the ManLoOut and ManHiOut output parameters to the LoScale and HiScale input parameters of the assigned analog output channel block. Caution: If the valves have an open neutral position (SafePos = 1), this interconnection must be crossed over: LoScale = ManHiOut and HiScale = ManLoOut. On the controller, 0% is always interpreted as the valve being closed and 100% as the valve being open, but the channel block then issues a control signal of 0% for a 100% controller manipulated variable.

### Tracking and limiting a manipulated variable

The block provides the standard function Tracking and limiting a manipulated variable (Page 222).

In case of a large control deviation, when disabling the tracking or forced tracking function and switching to the automatic mode, it may happen that the integral action is set far outside the manipulated variable limits. See Feature bit With accelerated return of the integral action from the manipulated variable limit (Page 193).

Tracking has a higher priority than interlock in this block.

### Neutral position

The block provides the standard function Neutral position for motors, valves and controllers (Page 54).

### Group error

This block provides the standard function Outputting group errors (Page 130).

The following parameters are taken into consideration when forming the group error:

- CSF

### Outputting a signal for start readiness

This block provides the standard function Outputting a signal for start readiness (Page 59).

### "Actuator active" information

If the manipulated variable MV is greater than the minimum manual limit ManLoLim, this is recognized as actuator active. This status can be used to display a custom symbol in the process image, for example, and is stored in the status word (you can find additional information under "Status word" in the Description of PIDConR (Page 1094) section).

### Limit monitoring of position feedback

The block provides the standard function Limit monitoring of the feedback (Page 101).

### Group display SumMsgAct for limit monitoring, CSF and ExtMsgx

The block provides the standard function Group display for limit monitoring, CSF and ExtMsgx (Page 92).

### External/internal setpoint specification

This input of setpoints is carried out either using a CFC/SFC program or using the faceplate (operator). The operator can specify an internal setpoint value (`SP_Int`) or a higher-level open-loop control will specify an external setpoint value (`SP_Ext`).

#### Setpoint specification internally and externally using faceplate

With PIDConR, the setpoint signal source is selected via the faceplate at the SP_IntOp = 1 parameter for the internal setpoint specification SP_ExtOp = 1 for external setpoint specification, just as it is with other controllers.

**Note**

In contrast to the other controllers, with PIDConR you can only switch to external setpoint specification via SP_ExtOp = 1 in automatic mode or in the program SP mode.

#### Setpoint specification internally and externally using interconnection

With PIDConR the switchover between internal and external setpoint via interconnectable input parameters follows a different logic than that used for other controller blocks. This logic is oriented towards the special requirements of the US market.

The setpoint signal source (internal/external) can be selected by interconnection along with selection of the operating mode using the `AutIntSet` input parameter for automatic with an internal setpoint and `AutExtSet` for automatic with an external setpoint (additional information is available in the PIDConR modes (Page 1100) section). The commands issued by interconnection take priority over the entries in the faceplate, i.e. if such a command input is made, the associated operator controls are locked in the faceplate. Input parameters `SP_LiOp`, `SP_ExtLi` and `SP_IntLi` on PIDConR are not therefore needed.

The PIDConR also offers a special way of influencing the setpoint specification via interconnectable input parameters. Loading setpoints. If the `SP_LoadOn` = 1 input parameter is set, the controller goes into "automatic" mode. The value of the `SP_Load` input parameter is limited according to an internal setpoint and used for control purposes. If the parameter `SP_LoadOn` changes back from 1 to 0, the controller remains in "automatic" mode and sets the default setpoint back to the internal setpoint.

Loading a setpoint via `SP_LoadOn` takes priority over all other forms of setpoint specification.

### Bumpless switchover from external to internal setpoint

The parameter `SP_TrkExt` = 1 is used so that the internal setpoint tracks the external setpoint to achieve a bumpless switchover from the external to the internal setpoint. This allows unwanted jumps at the output parameter to be avoided.

### Bumpless switchover between internal and external setpoint with PID initialization

With `Feature.Bit14` With PID initialization if mode SP intern extern changes (Page 207)you can make a bumpless switchover between internal and external setpoint. During switchover the PID algorithm will be initialized.

### Setpoint limiting for external setpoints

The block provides the standard function Setpoint limiting for external setpoints (Page 222).

### Gradient limit of the setpoint

The block provides the standard function Gradient limit of the setpoint (Page 132).

### Using setpoint ramp

The block provides the standard function Using setpoint ramp (Page 131).

### Tracking setpoint in manual mode

The block provides the standard function Tracking setpoint in manual mode (Page 222).

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Process value (`SimPV`, `SimPV_Li`)
- Position feedback (`SimRbk`, `SimRbkLi`)

### Bypass function

This block provides the standard function Bypassing signals (Page 115).

### Limit monitoring of the process value

The block provides the standard function Limit monitoring of the process value (Page 93) with the alarm delay type Two time values for each individual limit (Page 228).

The PIDConR is the only block to have separate input parameters for the alarm delay at the high and low limits. Delay alarm for control deviation at setpoint step changes (Page 216)

### Providing limits at the output

For further connections to the other blocks, the following input parameters are also displayed with the corresponding output parameters:

- `PV_HysOut` := `PV_Hyst`
- `PV_AH_Out` := `PV_AH_Lim`
- `PV_WH_Out` := `PV_WH_Lim`
- `PV_TH_Out` := `PV_TH_Lim`
- `PV_TL_Out` := `PV_TL_Lim`
- `PV_WL_Out` := `PV_WL_Lim`
- `PV_AL_Out` := `PV_AL_Lim`
- `ER_AH_LimOut` := `ER_AH_Lim`
- `ER_AL_LimOut` := `ER_AL_Lim`

### Error signal generation and dead band

The block provides the standard function Error signal generation and dead band (Page 217).

### Delay alarm for control deviation at setpoint step changes

The block provides the standard function Delay alarm for control deviation at setpoint step changes (Page 216)

### Limit monitoring of error signal

The block provides the standard function Limit monitoring of setpoint, manipulated variable and control deviation (Page 102). The monitoring of error signal works with the alarm delay type Two time values per limit pair (Page 227). With the `Feature` Bit Separate delay times for each alarm (Page 181), the alarm delay type Two time values for each individual limit (Page 228) can be activated.

### Inverting control direction

The block provides the standard function Inverting control direction (Page 217).

### Physical standardization of setpoint, manipulated variable and process value

Controller gain `Gain` is entered either using a physical variable or as standardized value.

`Gain` as a physical variable [`MV_Unit` / `PV_Unit`]:

The standardized variables retain their default values:

- `NormPV.High` = 100 and `NormPV.Low` = 0
- `NormMV.High` = 100 and `NormMV.Low` = 0

The effective gain is:

`GainEff` = `Gain`

Entering a standardized `Gain` (dimensionless):

Change the standardized variables to the actual range of the process values and manipulated variables.

- Internal and external setpoints; the process value and corresponding parameters are entered according to the physical measuring range of the process value.
- The manual value, the tracking value of the manipulated variable, disturbance variable feedforward and the corresponding parameters are set according to the physical measuring range of the manipulated variable.

The effective gain is:

`GainEff` = (`NormMV.High` - `NormMV.Low`) / (`NormPV.High` - `NormPV.Low`) · `Gain`

### Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 240).

### PID algorithm

The manipulated variable is generated in automatic mode according to the following algorithm:

For more information refer to "How it works" in Description of PIDConR (Page 1094)

$$MV = G_{ainEff}\left[1+\frac{T_D s}{\frac{T_D}{D_{iffGain}}s+1}\right](SP-PV)+Reset$$

$$Reset = \left(\frac{1}{T_I s+1}\right) MV, \qquad \text{ExtRstOn = 0}$$

$$Reset = \left(\frac{1}{T_I s+1}\right) \text{ExtReset}, \qquad \text{ExtRstOn = 1}$$

Where:

s = complex number of the Laplace transformation

This formula describes a standard application where P, I and D component is activated

The D component delay is derived from `TD` / `DiffGain`.

- The P component can be deactivated by `PropSel` = 0 .
- The I component can be deactivated by `IntSel` = 0 . The `MV_Offset` input parameter can then be used to add a constant value to the manipulated variable. Select this value such that the remaining control deviation equals zero at least at the control loop's typical operating point.
- The D component can be deactivated by `TD` = 0 or `DiffSel` = `1`.

**Note**

The formula describes a standard application where P, I and D components are activated and the D component is in the feedback circuit, while the P component is formed by the error signal (`PropSel` = `1`, `TI` <> `0` and `IntSel` = `1`, `TD` <> `0` and `DiffSel` = `1`, `DiffToFbk` = `1`, and `PropFacSP` = `1`).

If you set the `Feature` Bit Enabling bumpless change to the proportional gain, derivative time and amplification of the differentiator (Page 183) to 1, changes in the proportional gain `Gain` are carried out in a bumpless manner in "automatic" mode by converting the internal reset.

If the GainSched block is linked to the controller, you have to make the parameter settings for the `Feature` Bit Enabling bumpless change to the proportional gain, derivative time and amplification of the differentiator (Page 183) with 0.

The PID core algorithm is implemented in the PIDKernR function block which in turn calls a series of auxiliary functions for the 64-bit arithmetic. Every edge transition on the `InitPid` input parameter forces initialization equations to be carried out by PIDKernR. In so doing, the internal reset is calculated such that the `MV` output does not suffer a P step.

During initialization in the automatic mode with a large control deviation, it may happen that the integral action is set far outside the manipulated variable limits. See `Feature` bit With accelerated return of the integral action from the manipulated variable limit (Page 193).

**Note**

The meaning of the controller parameters with PIDConR is different to that with the other PID controllers for all controller parameter settings with a D component. If you want to transfer parameter values from one kind of controller to another, they have to be converted using the following formulas. The calculation is based on an ideal transfer function without a delay in the D component, and all three control channels are applied to the `ER` = `SP` - `PV` control deviation.

Parallel controller structure (for example, PIDConL):

$$MV = G_{ainEff}\left(1+\frac{1}{T_I s}+T_D s\right)ER$$

The parameters of the serial-interactive controller structure (PIDConR):

$$MV = G'_{ainEff}\left(1+T'_D s\right)\left(1+\frac{1}{T'_I s}\right)ER$$

These parameters are marked using a speech mark. Both controllers calculate the same manipulated variable if the parameter values of the serial-interactive structure are determined by the following substitution:

$$G'_{ainEff} = \alpha G_{ainEff} \ , \quad T'_I = \alpha T_I \ , \quad T'_D = \frac{1}{\alpha} T_D$$

where $\alpha$ is the conversion factor:

$$\alpha = \frac{1}{2} \pm \sqrt{\frac{1}{4} - \frac{T_D}{T_I}}$$

The argument of the square root is negative and the conversion is impossible if:

$$\frac{T_D}{T_I} > \frac{1}{4}$$

In such cases, the output of PIDConL cannot be exactly replicated by PIDConR. In all other cases, there are two solutions for $\alpha$ and therefore two parameter sets that will generate the same control action of the ideal controller structures. The conversion in the other direction, from serial-interactive to parallel controller structure is always possible by:

$$\alpha = \frac{T'_I}{T'_I + T'_D}$$

You can see that the conversion factor only equals one if $T_D = 0$.

In PID-Tuner, the conversion for ideal controller structures is performed automatically before downloading of parameters to the PIDConR function block. The first of the two possible solutions (plus square root) is used. If the argument of the square root is negative, it will be set to zero, which implies that PIDConR control performance may deteriorate slightly.

### Structure segmentation at controllers

The block provides the standard function Structure segmentation at controllers (Page 224).

### Use output point for the manipulated variable calculation (external reset)

For the `ExtResOn` = 0 input parameter (default setting), the starting point for the manipulated variable calculation within the block is taken from manipulated variable `MV`. In other words, this is the manipulated variable of the last sampling step. If `ExtResOn` = 1, the `ExtReset` start parameter is used. This is used in particular for networked control structures, such as cascade or transfer control.

### Anti-windup

A controller with incremental algorithm (external reset) inherently has an anti-windup reaction because the starting point for the manipulated variable calculation (external reset value) is limited provided it is taken internally from manipulated variable `MV` or is taken from another signal source with limitation. If the starting point for the manipulated variable calculation (external reset value) is at the limit (`MV_HiLim` or `MV_LoLim`), the I component is automatically frozen.

### Feedforwarding and limiting disturbance variables

The block provides the standard function Feedforwarding and limiting disturbance variables (Page 223).

### Control loop is open (`CascaCut`)

The `CascaCut` output indicates whether the control loop within this block is open. In the open control loop, the master controller must be set to tracking mode to avoid until windup behavior. If you use this block in a cascade structure, then connect the output `CascaCut` to the input `MV_TrkOn` of the master controller.

`CascaCut` is active in the following cases:

- Block is not in automatic mode
- `MV` program mode is active
- Setpoint is not in external mode
- Interlock is active

### Forming the signal status for blocks

The block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

- Signal status for the process value `PV_Out`:
  The signal status of the output parameter `PV_Out` always corresponds to the signal status of input parameter `PV` or, if the block is in simulation mode, 16#60.
- Signal status for the setpoint value `SP`:
  The signal status of the SP output parameter is always equivalent to the signal status of input parameter `SP_Ext` or `SP_Int`, depending on how the setpoint is specified. If the internal setpoint `SP_Int` is used, the signal status is always output as 16#80.
- Signal status of the error signal `ER`:
  The signal status of output parameter `ER` is obtained from the worst signal status of the two output parameters `PV_Out` and `SP` and is output. The signal status `16#60` (external simulation) is suppressed because the block acts as a sink with external simulation.
- Signal status for the manipulated variable `MV`:
  The signal status of output parameter `MV` is obtained in "automatic mode" or in "program mode" with default setpoint from the worst signal status of the two parameters `FFwd` and `ER` and is output. In "manual mode", the signal status is output as good. The signal status `16#60` (external simulation) is suppressed because the block acts as a sink with external simulation. In "manual mode", the signal status is output as good.

- Signal status for position feedback `RbkOut`:
  The signal status of `RbkOut` always corresponds to the signal status of input parameter `Rbk` or, if the block is in simulation mode, `16#60`.
- Worst signal status:
  The worst signal status `ST_Worst` for the block is formed from the following parameters:
  - `SP.ST`;
  - `PV_Out.ST`;
  - `FFwd.ST`;
  - `RbkOut.ST`;
  - `MV_ChnST.ST`;

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 5 | Changes signal status of outputs in OOS (Page 208) |
| 6 | Ramp rate calculation (Page 191) |
| 8 | Separate delay times for each alarm (Page 181) |
| 9 | Substitution value is active if the block is in bypass (Page 199) |
| 11 | Gradient limitation with time duration (Page 197) |
| 14 | With PID initialization if mode SP intern extern changes (Page 207) |
| 15 | Neutral position manipulated variable takes effect with "out of service" operating mode (Page 177) |
| 16 | Neutral position manipulated variable takes effect at startup (Page 176) |
| 17 | With accelerated return of the integral action from the manipulated variable limit (Page 193) |
| 19 | Enabling program mode (Page 170) |
| 20 | Enabling bumpless change to the proportional gain, derivative time and amplification of the differentiator (Page 183) |
| 21 | Switching operator controls for external setpoint to visible (Page 154) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 23 | SP following PV in open loop has no priority over SP_Ext and SP limits (Page 193) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 28 | Disabling operating points (Page 156) |
| 29 | Signaling limit violation (Page 181) |
| 30 | Dead band is temporarily disabled (Page 152) |
| 31 | Limits disabled in faceplate (Page 206) |

### Configurable reactions using the `Feature2` parameter

| Bit | Function |
|---|---|
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) |
| 11 | Suppress MsgLock and "Out of service" mode for a connected message block (Page 202) |
| 12 | Disable limit act output when limit value message is disabled (Page 209) |

### Operator permissions

The block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" `AutModOp` |
| 1 | 1 = Operator can switch to "manual mode" `ManModOp` |
| 2 | 1 = Operator can switch to "Out of service" mode `OosOp` |
| 3 | 1 = Operator can switch to "program mode" `AdvCoOn` |
| 4 | 1 = Operator can switch the setpoint to "external" `SP_ExtOp` |
| 5 | 1 = Operator can switch the setpoint to "internal" `SP_IntOp` |
| 6 | 1 = Operator can change the internal setpoint `SP_Int` |
| 7 | 1 = Operator can change the manual parameter `Man` |
| 8 | 1 = Operator can change operation high limit of the setpoint `SP_InHiLim` |
| 9 | 1 = Operator can change operation low limit of the setpoint `SP_InLoLim` |
| 10 | 1 = Operator can change the operation high limit of the manipulated variable `ManHiLim` |
| 11 | 1 = Operator can change the operation low limit of the manipulated variable `ManLoLim` |
| 12 | 1 = Operator can enable the setpoint's gradient limitation function `SP_RateOn` |
| 13 | 1 = Operator can raise the gradient limit `SP_UpRaLim` of the setpoint |
| 14 | 1 = Operator can lower the gradient limit `SP_DnRaLim` of the setpoint |
| 15 | 1 = Operator can switch between the time value or the gradient value for specifying the ramp `SP_RmpModTime` |
| 16 | 1 = Operator can change the ramp time `SP_RmpTime` |
| 17 | 1 = Operator can change the target setpoint `SP_RmpTarget` for the setpoint ramp |
| 18 | 1 = Operator can enable the setpoint ramp function `SP_RmpOn` |
| 19 | 1 = Operator can permit the PID optimization function `OptimEn` |
| 20 | 1 = Operator can enable the track setpoint in manual mode function `SP_TrkPV` |
| 21 | 1 = Operator can enable the bumpless switchover from external to internal `SP_TrkExt` |
| 22 | 1 = Operator can change the gain parameter `Gain` |
| 23 | 1 = Operator can change the integral time parameter `TI` |
| 24 | 1 = Operator can change the derivative time parameter `TD` |
| 25 | 1 = Operator can change the derivative gain parameter `DiffGain` |
| 26 | 1 = Operator can change the dead band parameter `DeadBand` |
| 27 | Not used |

| Bit | Function |
|---|---|
| 28 | 1 = Operator can change the derivative gain parameter ER_AH_DF |
| 29 | 1 = Operator can change the derivative gain parameter ER_AL_DF |
| 30 - 31 | Not used |

The block has the following permissions for the OS1Perm parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can change the limit (process value) PV_AH_Lim for the high alarm |
| 1 | 1 = Operator can change the limit (process value) PV_WH_Lim for the high warning |
| 2 | 1 = Operator can change the limit (process value) PV_TH_Lim for the high tolerance |
| 3 | 1 = Operator can change the hysteresis (process value) PV_Hyst |
| 4 | 1 = Operator can change the limit (process value) PV_TL_Lim for the low tolerance |
| 5 | 1 = Operator can change the limit (process value) PV_WL_Lim for the low warning |
| 6 | 1 = Operator can change the limit (process value) PV_AL_Lim for the low alarm |
| 7 | 1 = Operator can change the limit (error signal) ER_AH_Lim for the high alarm |
| 8 | 1 = Operator can change the hysteresis (error signal) ER_Hyst |
| 9 | 1 = Operator can change the limit (error signal) ER_AL_Lim for the low alarm |
| 10 | 1 = Operator can change the limit (position feedback) RbkWH_Lim for the high warning |
| 11 | 1 = Operator can change the hysteresis (position feedback) RbkHyst |
| 12 | 1 = Operator can change the limit (position feedback) RbkWL_Lim for the low warning |
| 13 | Not Used |
| 14 | 1 = Operator can activate bypass functionality |
| 15 | 1 = Operator can deactivate bypass functionality |
| 16 | 1 = Operator can activate the Simulation function SimOn |
| 17 | 1 = Operator can activate the Release for maintenance function MS_RelOp |
| 18 | 1 = Operator can activate / deactivate messages via PV_AH_MsgEn |
| 19 | 1 = Operator can activate / deactivate messages via PV_WH_MsgEn |
| 20 | 1 = Operator can activate / deactivate messages via PV_TH_MsgEn |
| 21 | 1 = Operator can activate / deactivate messages via PV_TL_MsgEn |
| 22 | 1 = Operator can activate / deactivate messages via PV_WL_MsgEn |
| 23 | 1 = Operator can activate / deactivate messages via PV_AL_MsgEn |
| 24 | 1 = Operator can activate / deactivate messages via ER_AH_MsgEn |
| 25 | 1 = Operator can activate / deactivate messages via ER_AL_MsgEn |
| 26 | 1 = Operator can activate / deactivate messages via RbkWH_MsgEn |
| 27 | 1 = Operator can activate / deactivate messages via RbkWL_MsgEn |
| 28 | 1 = Operator can change the simulation value SimPV |
| 29 | 1 = Operator can change the simulation value SimRbk |
| 30 | 1 = Operator can activate the derivative action to the feedback path DiffToFbk |
| 31 | 1 = Operator can change the proportional action to the feedback path PropFacSP |

The block has the following permissions for the OS2Perm parameter:

| Bit | Function |
|---|---|
| 0 - 15 | reserved |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Interlocks

This block provides the following interlocks:

- Interlock without reset ("Interlock")

You can find additional information on this in the section Interlocks (Page 107).

### Disabling interlocks

This block provides the standard function Disabling interlocks (Page 111).

### Release for maintenance

The block provides the standard function Release for maintenance (Page 70).

### Generating instance-specific messages

The block provides the standard function Generating instance-specific messages (Page 230) without the time stamp function in the I/O.

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Button labels

This block provides the standard function Labeling of buttons and text (Page 235).

Instance-specific text can be configured for the following parameters:

- AutModOp
- ManModOp
- AdvCoOn
- OosOp
- SP_ExtOp
- SP_IntOp

### Specifying the display area for process and setpoint values as well as operations

This block provides the standard function Display and operator input area for process values and setpoints (Page 233).

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### See also

PIDConR I/Os (Page 1121)

PIDConR messaging (Page 1118)

PIDConR error handling (Page 1117)

Program mode for controllers (Page 85)

EventTs functions (Page 2046)

## 6.9.4 PIDConR error handling

### Error handling of PIDConR

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Control system fault (CSF)

### Overview of error numbers

The `ErrorNum` output parameter can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. |

| Error number | Meaning of the error number |
|---|---|
| 30 | The value of PV can no longer be displayed in the REAL number field. |
| 31 | The value of SP_Ext can no longer be displayed in the REAL number field. |
| 32 | The value of FFwd can no longer be displayed in the REAL number field. |
| 33 | The value of MV_Trk can no longer be displayed in the REAL number field. |
| 34 | The value of MV_Forced can no longer be displayed in the REAL number field. |
| 35 | The value of Rbk can no longer be displayed in the REAL number field. |
| 36 | The value of MV can no longer be displayed in the REAL number field. |
| 51 | AutModLi = 1 and ManModLi = 1<br>SP_LiOp = 1 and SP_IntLi = 1 and SP_ExtLi = 1 |
| 52 | Feature.Bit30 = 1 and MeanMV.ST = 16#78 |
| 59 | = 1 "Gain is negative" |
| 60 | \|TI\| < SampleTime / 2 |
| 61 | \|TD\| < SampleTime |
| 62 | DiffGain < 1 or DiffGain > 10 |
| 63 | TD / DiffGain < SampleTime / 2 |
| 64 | PropFacSP < 0 or PropFacSP > 1 |
| 66 | NormPV_High = NormPV_Low |

### Control system fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 128) section for more on this.

### See also

PIDConR block diagram (Page 1140)

PIDConR I/Os (Page 1121)

PIDConR messaging (Page 1118)

PIDConR functions (Page 1103)

PIDConR modes (Page 1100)

Description of PIDConR (Page 1094)

## 6.9.5 PIDConR messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Process messages
- Instance-specific messages

### Process control fault

The following control system error messages can be output:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId2 | SIG 6 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If it changes to CSF = 1, a process control fault is triggered (MsgEvId2, SIG 6).

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 1 | Alarm - high | $$BlockComment$$<br>PV - high alarm limit violated |
| | SIG 2 | Warning - high | $$BlockComment$$<br>PV - high warning limit violated |
| | SIG 3 | Tolerance - high | $$BlockComment$$<br>PV - high tolerance limit violated |
| | SIG 4 | Tolerance - low | $$BlockComment$$<br>PV - low tolerance limit violated |
| | SIG 5 | Warning - low | $$BlockComment$$<br>PV - low warning limit violated |
| | SIG 6 | Alarm - low | $$BlockComment$$<br>PV - low alarm limit violated |
| | SIG 7 | Alarm - high | $$BlockComment$$<br>ER - high alarm limit violated |
| | SIG 8 | Alarm - low | $$BlockComment$$<br>ER - low alarm limit violated |
| MsgEvId2 | SIG 7 | Warning - high | $$BlockComment$$<br>Rbk - high warning limit violated |
| | SIG 8 | Warning - low | $$BlockComment$$<br>Rbk - low warning limit violated |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Instance-specific messages

You can use up to four instance-specific messages with this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId2 | SIG 1 | AS process control message - fault | $$BlockComment$$<br>External message 1<br>Status 16#@5%x@ |
| | SIG 2 | AS process control message - fault | $$BlockComment$$<br>External message 2<br>Status 16#@6%x@ |
| | SIG 3 | AS process control message - fault | $$BlockComment$$<br>External message 3<br>Status 16#@7%x@ |
| | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External message 4<br>Status 16#@8%x@ |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance **MsgEvId1**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |
| 4 | Process value PV_Out |
| 5 | Control deviation ER |
| 6 | ExtVa106 |
| 7 | ExtVa107 |
| 8 | Not allocated |
| 9 | Not allocated |
| 10 | Not allocated |

The associated values 6 ... 7 are allocated to the parameters ExtVa106 ... ExtVa107 and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### Associated values for message instance **MsgEvId2**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |

| Associated value | Block parameters |
|---|---|
| 4 | Position feedback RbkOut |
| 5 | Signal status ExtMsg1 |
| 6 | Signal status ExtMsg2 |
| 7 | Signal status ExtMsg3 |
| 8 | Signal status ExtMsg4 |
| 9 | ExtVa209 |
| 10 | ExtVa210 |

The associated values 9 ... 10 are allocated to the parameters ExtVa209 ... ExtVa210 and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

PIDConR block diagram (Page 1140)

PIDConR I/Os (Page 1121)

PIDConR error handling (Page 1117)

PIDConR functions (Page 1103)

PIDConR modes (Page 1100)

Description of PIDConR (Page 1094)

## 6.9.6 PIDConR I/Os

I/Os of PIDConR

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AdvCoEn | 1 = Enable "program mode" via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AdvCoOn* | 1 = Enable "program mode" via faceplate.<br>The parameter is available on the WinCC OPC server to be read by external APC programs. | BOOL | 0 |
| AdvCoModSP | Type of "program mode":<br>1 = Setpoint specification<br>0 = Manipulated variable specification | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| AdvCoMstrOn | 1 = Enable (0-1) or disable (1-0) "program mode" via edge transition | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AdvCoMV | Specified value from the external program.<br>The parameter is available on the WinCC OPC server to be read by external APC programs. | REAL | 0.0 |
| AutExtSet* | 1 = Activate "automatic" mode with external setpoint by interconnection (SP = SP_Ext) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutIntSet* | 1 = Activate "automatic" mode with internal setpoint by interconnection (SP = SP_Op).<br>AutIntSet has higher priority thanAutExtSet. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutModOp* | 1 = "Automatic" mode via operator | BOOL | 0 |
| BatchEn | 1 = Enable allocation for batch control | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkTypeData* | This input can not be edit or linked and is for internal usage only! | | |
| BypProt | 1 = Interlock bypass during simulation | BOOL | 0 |
| BypLiOp | 1 = Bypass commands via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypLock | 1 = Bypass activation or deactivation is locked for operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypPV | Substitution value if block is in bypass | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| BypPVLi | 1 = Select bypass PV (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypPVOp | 1 = Select bypass PV (via operator) | BOOL | 0 |
| CPI_In | Input for control performance index, which is calculated by the assigned ConPerMon block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#78 |
| CSF | 1 = External error (control system fault)<br>Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DeadBand | Width of dead band | REAL | 0.0 |
| DiffGain | Gain of differentiator<br>[1..10] DiffGain = TD / (delay time of D component) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 5.0<br>• 16#80 |
| DiffSel | 1 = D component activated | BOOL | 1 |
| DiffToFbk* | 1 = D component is placed in the feedback | BOOL | 1 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| ER_A_DC* | Delay time for incoming ER high/low or only low alarms [s] | REAL | 0.0 |
| ER_AH_DC* | Delay time for incoming ER high alarms [s] | REAL | 0.0 |
| ER_AH_DFac* | Delay factor at positive setpoint step changes for incoming alarms at the error signal monitoring ER_AH_Lim | REAL | 0.0 |
| ER_A_DG* | Delay time for outgoing ER high/low or only low alarms [s] | REAL | 0.0 |
| ER_AH_DG* | Delay time for outgoing ER high alarms [s] | REAL | 0.0 |
| ER_AH_En | 1 = Activate alarm (high) for error signal monitoring | BOOL | 1 |
| ER_AH_Lim | Alarm limit (high) for error signal monitoring | REAL | 100.0 |
| ER_AH_MsgEn | 1 = Activate messages for alarm (high) for error signal monitoring | BOOL | 1 |
| ER_AL_DFac* | Delay factor at negative setpoint step changes for incoming alarms at the error signal monitoring ER_AL_Lim | REAL | 0.0 |
| ER_AL_En | 1 = Activate alarm (low) for error signal monitoring | BOOL | 1 |
| ER_AL_Lim | Alarm limit (low) for error signal monitoring | REAL | -100.0 |
| ER_AL_MsgEn | 1 = Activate messages for alarm (low) for error signal monitoring | BOOL | 1 |
| ER_Hyst | Alarm hysteresis for error signal | REAL | 1.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| EventTsIn | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| ExtMsg1 | 1 = Binary input for freely selectable message 1 is used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | 1 = Binary input for freely selectable message 2 is used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg3 | 1 = Binary input for freely selectable message 3 is used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg4 | 1 = Binary input for freely selectable message 4 is used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtReset | Value to which a reset is made if ExtRstOn = 1. | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ExtRstOn | 1 = Reset externally | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| ExtVa209 | Associated value 9 for messages (MsgEvID2) | ANY | |
| ExtVa210 | Associated value 10 for messages (MsgEvID2) | ANY | |
| Feature | I/O for additional functions (Page 1103) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 20: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 0<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Feature2 | I/O for additional functions (Page 1103) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| FFwd* | Input for additive disturbance variable activation | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| FFwdHiLim | Limit (high) for additive disturbance variable activation | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| FFwdLoLim | Limit (low) for additive disturbance variable activation | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• -100.0<br>• 16#80 |
| Gain | Proportional gain<br>Gain.ST = 16#FF: Enabled in face-plate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#FF |
| InitPid* | InitPid edge transitions result in the PID algorithm's initialization equations being carried out. Is used, for example, for SP changes without MV jumps | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Intl_En | 1 = Interlock without reset (interlock, Intlock parameter) is in effect | BOOL | 1 |
| Intlock | 0 = Interlock without reset is in effect<br>Once the interlock condition has cleared, the block can be operated without reset<br>1 = Interlock not activated | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| IntSel | 1 = I component activated | BOOL | 1 |
| Man* | Manual specification for the manipulated variable | REAL | 0.0 |
| ManHiLim | Limit (high) for manual parameter Man | REAL | 100.0 |
| ManLoLim | Limit (low) for manual parameter Man | REAL | 0.0 |
| ManModOp* | 1 = "Manual" mode via OS operator | BOOL | 1 |
| ManSet* | 1 = Activate "manual" mode via interconnection.<br>ManSet has higher priority thanAutIntSet. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_RelOp* | 1 = Release for maintenance by OS operator | BOOL | 0 |
| MsgEvID1 | Message number (assigned automatically) | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MsgEvID2 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_Bump* | Default value for controller output MV if MV_BumpOn = 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_BumpOn | 1 = Set controller output MV:= MV_Bump without (!) taking the controller into "manual" mode. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_ChnST | Signal status of output channel for MV<br>Should be connected to an output channel block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| MV_Close | 1 = Close adjustment valve by interconnection, i.e. MV:= MV_LoLim<br>MV_Close has higher priority than MV_Open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_Forced* | Forced manipulated variable that is not limited and assumes top priority | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_ForOn | 1 = Output forced manipulated variable MV_Forced unlimited at output MV | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_HiLim | Limit (high) for manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| MV_Mean | Mean value of the MV in the time window | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#78 |
| MV_LoLim | Limit (low) for manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_Offset | Manipulated variable for ER=0, operating point for controller with deactivated I component | REAL | 0.0 |
| MV_Open | 1 = Open adjustment valve by interconnection, i.e. MV:= MV_HiLim<br>MV_Open has higher priority than MV_TrkOn | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MV_OpScale | OS display range for manipulated variable MV | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| MV_SafePos | Manipulated variable neutral position | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_Trk* | Tracking value for the manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_TrkOn | 1 = Tracking of manipulated variable MV | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_Unit | Unit of measure for manipulated variable | INT | 1342 |
| NegGain | 0 = Effective proportional gain GainEff is positive<br>1 = Effective proportional gain GainEff is negative | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NormMV | Manipulated variable range (MV) for standardizing the proportional gain (GAIN) | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| NormPV* | Process value range (PV) for standardizing the proportional gain (GAIN) | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| Occupied | Occupied by batch control | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OptimEn* | 1 = Enable optimization of PID parameters by PID tuner | BOOL | 0 |
| OptimOcc* | 1 = Optimization running | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 1103) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OS1Perm | I/O for operator permissions (Page 1103) | STRUCT<br>• Bit 0: BOOL<br>• Bit 18: BOOL<br>• Bit 19:BOOL<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1<br>• 1 |
| OS2Perm | I/O for operator permissions | STRUCT<br>• Bit 0: BOOL<br>• Bit 15: BOOL | -<br>• 1<br>• 1 |
| PropFacSP | Applying the P component to the feedback [0..1].<br>0 = P component fully in feedback | REAL | 1.0 |
| PropSel* | 1 = Activate P component | BOOL | 1 |
| PV* | Process value (controlled variable) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_Unit | Unit of measure for process value | INT | 1001 |
| PV_AH_DC* | Delay time for incoming PV high alarms [s] | REAL | 0.0 |
| PV_AH_DG* | Delay time for outgoing PV high alarms [s] | REAL | 0.0 |
| PV_AH_En | 1 = Enable PV alarm limit (high) | BOOL | 1 |
| PV_AH_Lim | Limit PV alarm (high) | REAL | 95.0 |
| PV_AH_MsgEn | 1 = Enable PV alarm (high) message | BOOL | 1 |
| PV_AL_En | 1 = Enable PV alarm limit (low) | BOOL | 1 |
| PV_AL_Lim | PV alarm limit (low) | REAL | 5.0 |
| PV_AL_MsgEn | 1 = Enable PV alarm (low) message | BOOL | 1 |
| PV_AL_DC* | Delay time for incoming PVlow alarms [s] | REAL | 0.0 |
| PV_AL_DG* | Delay time for outgoing PV low alarms [s] | REAL | 0.0 |
| PV_Hyst | Hysteresis for PV alarm, warning and tolerance limits | REAL | 1.0 |
| PV_OpScale | Limit for scale in PV bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| PV_TH_DC* | Delay time for incoming PV high tolerance messages [s] | REAL | 0.0 |
| PV_TH_DG* | Delay time for outgoing PV high tolerance messages [s] | REAL | 0.0 |
| PV_TH_En | 1 = Enable PV tolerance limit (high) | BOOL | 0 |
| PV_TH_Lim | Limit PV tolerance message (high) | REAL | 85.0 |
| PV_TH_MsgEn | 1 = Enable message for PV tolerance message (high) | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_TL_DC* | Delay time for incoming PV low tolerance messages [s] | REAL | 0.0 |
| PV_TL_DG* | Delay time for outgoing PV low tolerance messages [s] | REAL | 0.0 |
| PV_TL_En | 1 = Enable PV tolerance limit (low) | BOOL | 0 |
| PV_TL_Lim | Limit PV tolerance message (low) | REAL | 15.0 |
| PV_TL_MsgEn | 1 = Activate message for PV tolerance message (low) | BOOL | 1 |
| PV_WH_DC* | Delay time for incoming PV high warnings [s] | REAL | 0.0 |
| PV_WH_DG* | Delay time for outgoing PV high warnings [s] | REAL | 0.0 |
| PV_WH_En | 1 = Enable PV warning limit (high) | BOOL | 1 |
| PV_WH_Lim | Limit PV warning (high) | REAL | 90.0 |
| PV_WH_MsgEn | 1 = Enable PV warning (high) message | BOOL | 1 |
| PV_WL_DC* | Delay time for incoming PV low warnings [s] | REAL | 0.0 |
| PV_WL_DG* | Delay time for outgoing PV low warnings [s] | REAL | 0.0 |
| PV_WL_En | 1 = Enable PV warning limit (low) | BOOL | 1 |
| PV_WL_Lim | Limit PV warning (low) | REAL | 10.0 |
| PV_WL_MsgEn | 1 = Enable PV warning (low) message | BOOL | 1 |
| Rbk* | Position feedback for display on OS | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| RbkHyst | Alarm hysteresis for position feedback | REAL | 1.0 |
| RbkW_DC* | Delay time for incoming Rbk warnings [s] | REAL | 0.0 |
| RbkW_DG* | Delay time for outgoing Rbk warnings [s] | REAL | 0.0 |
| RbkWH_En | 1 = Enable warning (high) for position feedback | BOOL | 1 |
| RbkWH_Lim | Limit for position feedback of warning (high) | REAL | 100.0 |
| RbkWH_MsgEn | 1 = Enable messages for warning (high) for position feedback | BOOL | 1 |
| RbkWL_En | 1 = Enable warning (low) for position feedback | BOOL | 1 |
| RbkWL_Lim | Limit for position feedback of warning (low) | REAL | 0.0 |
| RbkWL_MsgEn | 1 = Enable messages for warning (low) for position feedback | BOOL | 1 |
| RefStdDevIn | Reference value of PV standard deviation (sigma) in defined "good" state of control loop | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#78 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RstBypLi | 1 = Reset bypass PV (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstBypOp | 1 = Reset bypass PV (via operator) | BOOL | 0 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SafePos | 1 = Neutral position (Page 54) for controller manipulated variable is ManHiLim<br>0 = Neutral position for controller manipulated variable is ManLoLim | BOOL | 0 |
| SafePos2 | Neutral position for controller manipulated variable:<br>0 = SafePos is valid<br>1 = Neutral position is MV_SafePos<br>2 = Neutral position is last manipulated variable (stop) | INT | 0 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | - |
| SettliFactor | Factor to increase the settling time to adjust the dead band | REAL | 2.0 |
| SettliTime | Settling time [s] of the control loop determined by the ConPerMon block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#78 |
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SimPV* | Process value used for SimOn = 1 | REAL | 0.0 |
| SimPV_Li | Process value that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SimRbk* | Position feedback used for SimOn = 1 | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SimRbkLi | Position feedback used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_DnRaLim* | Limit (low) for the gradient of the setpoint [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 100.0 |
| SP_ExHiLim | Limit (high) for external setpoint | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| SP_ExLoLim | Limit (low) for external setpoint | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_Ext* | external setpoint - (to interconnection) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_ExtOp* | 1 = Select external setpoint (via operator) | BOOL | 0 |
| SP_InHiLim | Limit (high) of internal setpoint | REAL | 100.0 |
| SP_InLoLim | Limit (low) of internal setpoint | REAL | 0.0 |
| SP_Int* | Internal setpoint for operation | REAL | 0.0 |
| SP_IntOp* | 1 = Select internal setpoint (via operator) | BOOL | 0 |
| SP_LiDnRaLim | Linkable limit (low) for the gradient of the setpoint [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_LiRmpTime | Linkable time [Time unit] for gradient limitation in ramp mode duration<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_LiUpRaLim | Linkable gradient limit (high) for the setpoint [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_Load* | Defined setpoint, if SP_LoadOn = 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_LoadOn* | 1 = Take controller into "automatic mode" with internal setpoint and set setpoint at SP:= SP_Load.<br>This kind of setpoint specification has maximum priority. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_RateOn* | 1 = Activate limitation of setpoint gradients | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP_RmpModTime | 1 = Use time values (SP_RmpTime, SP_LiRmpTime)<br>0 = Use gradient limits (SP_DnRaLim, SP_LiDnRaLim, SP_UpRaLim, SP_LiUpRaLim) | BOOL | 0 |
| SP_RmpOn* | 1 = Activate setpoint ramp to target setpoint SP_RmpTarget | BOOL | 0 |
| SP_RmpTarget* | Target setpoint for setpoint ramp | REAL | 0.0 |
| SP_RmpTime* | Time [Time unit] for setpoint ramp from current SP up to SP_RmpTarget or time [Time unit] for gradient limitation in ramp mode duration<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_TrkExt | 1 = Bumpless switchover from external to internal setpoint active | BOOL | 1 |
| SP_TrkPV | 1 = Setpoint follows PV in "manual mode" and with tracking | BOOL | 0 |
| SP_UpRaLim* | Gradient limit (high) for the setpoint [Unit/(Time unit)]Time unit depends on TimeFactor | REAL | 100.0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| TD | Derivative time [s]<br>TD.ST = 16#FF: Enabled in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| TI | Integral time [s]<br>TI.ST = 16#FF: Enabled in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#FF |
| TimeFactor | Time unit:<br>0 = Seconds<br>1 = Minutes<br>2 = Hours | INT | 0 |
| UA1unit | Unit of measure for analog auxiliary value 1 | INT | 0 |
| UA2unit | Unit of measure for analog auxiliary value 2 | INT | 0 |
| UserAna1 | Analog auxiliary value 1 | STRUCT<br>Value: REAL<br>ST: BYTE | <br>0.0<br>16#FF |
| UserAna2 | Analog auxiliary value 2 | STRUCT<br>Value: REAL<br>ST: BYTE | <br>0.0<br>16#FF |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AdvCoAct | 1 = "Program mode" enabled.<br>The parameter is available on the WinCC OPC server to be read by external APC programs. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AdvCoRdy | 1 = "Program mode" available | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutAct | 1 = "Automatic mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypassAct | 1 = Bypass is activated in this block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CascaCut | Cascade connection:<br>1 = Control chain from primary to secondary controller is interrupted | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| DynDeadBand | Dynamic dead band | REAL | 0.0 |
| PhaseDeadBand | Phase for the dynamic adjustment of the dead band<br>0: Dead band enabled<br>1: Dead band disabled<br>2: Settling time | INT | 0 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ER | Error signal | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ER_A_DCOut | Effective delay time [s] for incoming alarms at the error signal monitoring | REAL | 0.0 |
| ER_AH_Act | 1 = Alarm limit (high) for control deviation violated.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ER_AL_Act | 1 = Alarm limit (low) for control deviation violated.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| ER_AH_LimOut | ER_AH_Lim as output | REAL | 0.0 |
| ER_AL_LimOut | ER_AL_Lim as output | REAL | 0.0 |
| ErrorNum* | Output of pending error number.<br>For error numbers that can be output by this block, see PIDConR error handling (Page 1117) | INT | -1 |
| FFwdHiAct | 1 = Limit (high) for additive disturbance variable activation violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FFwdLoAct | 1 = Limit (low) for additive disturbance variable activation violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| GainEff | Effective proportional gain depends on NegGain, Gain, NormPV, and NormMV | REAL | 1.0 |
| GrpErr | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IntlockOut | Intlock as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LockAct | 1 = Interlock is in effect | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LoopClosed | 1 = Control loop closed<br>0 = Control loop open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManAct | 1 = "Manual mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| ManHiOut | Limit (high) for "manual mode", corresponds to input parameter ManHiLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| ManLoOut | Limit (low) for "manual mode", corresponds to input parameter ManLoLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MS_Release | Release for maintenance: 1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message acknowledgement status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |
| MsgAckn2 | Message acknowledgement status 2 (output ACK_STATE of second ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | Alarm error 1 (output ERROR of first ALARM_8P) | BOOL | 0 |
| MsgErr2 | Alarm error 2 (output ERROR of second ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| MsgStat2 | Message status 2 (output STATUS of second ALARM_8P) | WORD | 16#0000 |
| MV | Manipulated variable | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_HiAct | 1 = Limit (high) of manipulated variable violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_LoAct | 1 = Limit (low) of manipulated variable violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_UnitOut | Unit of measure for manipulated variable, for interconnecting to the MV_Unit input parameter of the ConPerMon block | INT | 0 |
| OosAct | 1 = Block is "out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1Perm | DWORD | 16#FFFFFFFF |
| OS1PermLog | Display of OS1Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS2PermOut | Display of OS2_Perm | WORD | 16#FFFF |
| OS2PermLog | Display of OS2_Perm with settings changed by the block algorithm | WORD | 16#FFFF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_AH_Act | 1 = PV alarm (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_AH_Out | PV - High alarm limit output | REAL | 0.0 |
| PV_AL_Act | 1 = PV alarm (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_AL_Out | PV - Low alarm limit output | REAL | 0.0 |
| PV_HysOut | PV - Alarm hysteresis output | REAL | 0.0 |
| PV_Out | Output for process value | STRUCT<br>• Value: REAL<br>• ST: BYTE<br>• Bit 1: BOOL of internal fill byte (Bypass information bit) | -<br>• 0.0<br>• 16#80<br>• 0 |
| PV_OpScaleOut | PV_OpScale as output | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| PV_TH_Act | 1 = PV tolerance message (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_TH_Out | PV - High tolerance limit output | REAL | 0.0 |
| PV_TL_Act | 1 = PV tolerance message (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_TL_Out | PV - Low tolerance limit output | REAL | 0.0 |
| PV_ToleHi | Limit (high) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_ToleLo | Limit (low) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_UnitOut | Unit of measure for process value, for interconnecting to the PV_Unit input parameter of the ConPerMon block | INT | 0 |
| PV_WH_Act | 1 = PV warning (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_WH_Out | PV - High warning limit output | REAL | 0.0 |
| PV_WL_Act | 1 = PV warning (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_WL_Out | PV - Low warning limit output | REAL | 0.0 |
| RbkOut | Output for position feedback | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RbkWH_Act | 1 = Warning (high) for position feedback active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RbkWL_Act | 1 = Warning (low) for position feedback active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToStart | 1 = Active start readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SettlingTimer | Settling time for a closed control loop | REAL | 0.0 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP | Setpoint used by controller | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_DnRaAct | 1 = Negative gradient limiting of setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExHiAct | 1 = Limit (high) for external setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExLoAct | 1 = Limit (low) for external setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtAct | 1 = External setpoint is active<br>0 = Internal setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtOut | External setpoint, corresponds to input parameter SP_Ext | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_InHiOut | Limit (high) for SP_Int corresponds to input parameter SP_InHiLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| SP_InLoOut | Limit (low) for SP_Int corresponds to input parameter SP_InLoLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_RateTarget | Target setpoint for the gradient limitation | REAL | 0.0 |
| SP_UpRaAct | Positive gradient limiting of setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_RemRT | Remaining ramp time of the setpoint | REAL | 0.0 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 1094) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1094) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1094) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 1094) | DWORD | 16#00000000 |
| SumMsgAct | 1 = Active hardware interrupt | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| UA1unitOut | UA1 as output | INT | 0 |
| UA2unitOut | UA2 as output | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| UserAna1Out | UserAna1 as output | STRUCT | |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| UserAna2Out | UserAna2 as output | STRUCT | |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Calculation of the output parameter ER_A_DCOut

ER_A_DC is assigned by default to the output before a setpoint change.

ER_A_DCOut = ER_A_DC

In the case of a setpoint change in the positive direction during automatic mode, the output is calculated as follows:

ER_A_DCOut = Maximum (ER_A_DC, ER_AH_DFac * Setpoint difference)

In the case of a setpoint change in the negative direction during automatic mode, the output is calculated as follows:

ER_A_DCOut = Maximum (ER_A_DC, -1*ER_AH_DFac * Setpoint difference)

When the control circuit has stabilized again, meaning

(ER_AL_Lim + ER_Hyst) <= ER <= (ER_AH_Lim – ER_Hyst)

and the delay time for outgoing alarms ER_A_DG has expired, the output is reset again to ER_A_DC: ER_A_DCOut = ER_A_DC

### Activating and deactivating the function:

The function is deactivated (default) when the following applies: ER_AH_DFac = 0.0 and ER_AL_DFac = 0.0

### See also

PIDConR block diagram (Page 1140)

PIDConR messaging (Page 1118)

PIDConR modes (Page 1100)

## 6.9.7 PIDConR block diagram

### PIDConRblock diagram

A block diagram is not provided for this block.

### See also

PIDConR I/Os (Page 1121)

PIDConR messaging (Page 1118)

PIDConR error handling (Page 1117)

PIDConR functions (Page 1103)

PIDConR modes (Page 1100)

Description of PIDConR (Page 1094)

## 6.9.8 Operator control and monitoring

### 6.9.8.1 PIDConR views

### Views of the PIDConR block

This block has the same views as the PIDConL block. Refer to the section Operator control and monitoring (Page 1060) of the PIDConL block.

# 6.10 PIDStepL - Step controller

## 6.10.1 Description of PIDStepL

### Object name (type + number) and family

Type + number: FB 1878

Family: Control

### Area of application for PIDStepL

The block is used for the following applications:

- Fixed setpoint control
- Cascade control
- Ratio control
- Split-range control
- Smith predictor closed-loop control
- Override control (override)

Note on restricting the area of application: The block is not intended for pulse-duration modulation. Examples of applications: Temperature control with electric heater, which can switched on an off via a (semiconductor) relay. The heating power required by the controller, for example 80%, is output in the form of binary pulses, whereby the duration of the on pulses in this case is four times as long as the pause duration. Such control is built with a combination of a continuous controller (such as PIDConL) and the pulse generator module PULSEGEN from the CFC library ELEM_400.

### How it works

The block is a PID step controller with binary output signals (manipulated variable signals). It is used to control integral action actuators (e.g. valves driven by motors).

The block functions following the PID algorithm with a delayed D component and an integrator with double precision.

The step controller can function both with and without a position feedback for the valve position.

The block is suitable for controlling sluggish control loops, for example, for temperatures and filling levels, and high-speed control loops, for example, for flow rates and speed. For a given CPU, a compromise has to be made between the number of controllers and the frequency with which the individual controllers have to be processed. The faster the modulated control loops are, i.e. the more frequently the manipulated variables have to be calculated per time unit, the lower the number of controllers that can be installed.

## Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

For the control loop monitoring to work as planned in the trend view of the controller faceplates, the

`S7_xarchive:='Value, shortterm;'`

attributes in the process tag types for control loops at the controller function block must be set for the following tags:

- Input parameters:
  - `CPI_In`
- Output parameters
  - `MV`
  - `MV_HiAct`
  - `MV_LoAct`
  - `LoopClosed`
  - `SP`
  - `PV_Out`
  - `PV_ToleHi`
  - `PV_ToleLo`

For the PIDStepL block, the Advanced Process Library contains templates for process tag types as examples and there is a example project (APL_Example_xx, xx designates the language variant) containing different application cases for this block.

Several process tag types are simulated in the example project and serve to explain the block function.

Examples of process tag types:

- Step controller with direct access to the actuator and without position feedback (StepControlDirect) (Page 2831)
- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 2832)

## Startup characteristics

Use the Feature Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section PIDStepL I/Os (Page 1163).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `AutAct.Value` |
| 6 | Invalid signal status |
| 7 | `ManAct.Value` |
| 8 | `SP_ExtAct.Value` |
| 9 | `MV_ForOn.Value` |
| 10 | `MV_TrkOn.Value` |
| 11 | NOT `RbkClosed.Value` |
| 12 | `Open.Value` |
| 13 | `Close.Value` |
| 14 | `Stop.Value` |
| 15 | `FbkOpened.Value` |
| 16 | `FbkClosed.Value` |
| 17 | `SimLiOp.Value` |
| 18 | `SimOn` AND `ManAct` |
| 19 | `AdvCoAct` |
| 20 | 1 = Input parameter `Rbk` is not interconnected (`RbkOut.ST` = 16#FF) |
| 21 | `NegGain` |
| 22 | 1 = SP ramp active |
| 23 | `OptimEn` |
| 24 | `OptimOcc` |
| 25 | Not used |
| 26 | Display of `BypassAct.Value` in faceplate (display and operator controls) and block icon |
| 27 | `AdvCoModSP` |
| 28 | 1 = Input parameter `OpenChnST` is interconnected |
| 29 | 1 = Input parameter `CloseChnST` is interconnected |
| 30 | 1 = Input parameter `StopChnST` is interconnected |
| 31 | `WithRbk` = 1 (Step controller with position feedback) |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `MsgLock` |
| 1 | `PV_AH_Act.Value` |
| 2 | `PV_WH_Act.Value` |

| Status bit | Parameter |
|---|---|
| 3 | PV_TH_Act.Value |
| 4 | PV_TL_Act.Value |
| 5 | PV_WL_Act.Value |
| 6 | PV_AL_Act.Value |
| 7 | PV_AH_En OR Feature.Bit31 |
| 8 | PV_WH_En OR Feature.Bit31 |
| 9 | PV_TH_En OR Feature.Bit31 |
| 10 | PV_TL_En OR Feature.Bit31 |
| 11 | PV_WL_En OR Feature.Bit31 |
| 12 | PV_AL_En OR Feature.Bit31 |
| 13 | PV_AH_MsgEn |
| 14 | PV_WH_MsgEn |
| 15 | PV_TH_MsgEn |
| 16 | PV_TL_MsgEn |
| 17 | PV_WL_MsgEn |
| 18 | PV_AL_MsgEn |
| 19 | ER_AH_Act.Value |
| 20 | ER_AL_Act.Value |
| 21 | ER_AH_En OR Feature.Bit31 |
| 22 | ER_AL_En OR Feature.Bit31 |
| 23 | ER_AH_MsgEn |
| 24 | ER_AL_MsgEn |
| 25 | RbkWH_Act.Value |
| 26 | RbkWL_Act.Value |
| 27 | RbkWH_En OR Feature.Bit31 |
| 28 | RbkWL_En OR Feature.Bit31 |
| 29 | RbkWH_MsgEn |
| 30 | RbkWL_MsgEn |
| 31 | MS_RelOp |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 | Effective signal 1 of the message block connected via EventTsIn |
| 1 | Effective signal 2 of the message block connected via EventTsIn |
| 2 | Effective signal 3 of the message block connected via EventTsIn |
| 3 | Effective signal 4 of the message block connected via EventTsIn |
| 4 | Effective signal 5 of the message block connected via EventTsIn |
| 5 | Effective signal 6 of the message block connected via EventTsIn |
| 6 | Effective signal 7 of the message block connected via EventTsIn |
| 7 | Effective signal 8 of the message block connected via EventTsIn |
| 8 | Delay of the PV_AH_Lim message |

| Status bit | Parameter |
|---|---|
| 9 | Delay of the PV_WH_Lim message |
| 10 | Delay of the PV_TH_Lim message |
| 11 | Delay of the PV_TL_Lim message |
| 12 | Delay of the PV_WL_Lim message |
| 13 | Delay of the PV_AL_Lim message |
| 14 | Delay of the ER_AH_Lim message |
| 15 | Delay of the ER_AL_Lim message |
| 16 | Collection of message delays |
| 17 | BypassAct.Value |
| 18 - 19 | Not used |
| 20 | 1= Suppressed messages |
| 21 - 26 | Not used |
| 27 | SP_UpRaAct, SP_DnRaAct limits enabled for gradient mode (SP_RateOn = 1) |
| 28 | GrpErr.Value |
| 29 | RdyToStart.Value |
| 30 | Not used |
| 31 | Internal error |

### Status word allocation for Status4 parameter

| Status bit | Parameter |
|---|---|
| 0 - 7 | Effective signal 9..16 of the message block connected via EventTsIn |
| 8 - 28 | Not used |
| 29 | 1 = Input parameter SP_LiRmpTime is not interconnected (SP_LiRmpTime.ST = 16#FF) |
| 30 | Auxiliary value 1 visible |
| 31 | Auxiliary value 2 visible |

### See also

PIDStepL functions (Page 1147)
PIDStepL messaging (Page 1160)
PIDStepL modes (Page 1146)
PIDStepL error handling (Page 1159)
PIDStepL block diagram (Page 1179)

## 6.10.2 PIDStepL modes

### PIDStepL modes

The block can be operated using the following modes:

- Automatic mode (Page 79)
- Manual mode (Page 79)
- Program mode for controllers (Page 85)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and bumpless switchover in the section Manual and automatic mode for control blocks (Page 79).

### "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the section Manual and automatic mode for control blocks (Page 79).

### "Program mode for controllers"

You can find general information about the "Program mode for controller" in the section Program mode for controllers (Page 85).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit5` =1 and `Feature.Bit5` =0 is as shown below:

| Output Parameters | `Feature.Bit5` =0 | `Feature.Bit5` =1 |
|---|---|---|
| Open.ST | 16#80 | 16#60 |
| Close.ST | 16#80 | 16#60 |
| Stop.ST | 16#80 | 16#60 |
| RbkOut.ST | 16#80 | 16#60 |
| RbkWH_Act.ST | 16#XX | 16#60 |
| RbkWL_Act.ST | 16#XX | 16#60 |
| SP_ExtAct.ST | 16#80 | 16#60 |
| SP.ST | 16#80 | 16#60 |
| SP_ExtOut.ST | 16#80 | 16#60 |

| Output Parameters | Feature.Bit5 =0 | Feature.Bit5 =1 |
|---|---|---|
| PV_Out.ST | 16#80 | 16#60 |
| PV_AH_Act.ST | 16#XX | 16#60 |
| PV_WH_Act.ST | 16#XX | 16#60 |
| PV_TH_Act.ST | 16#XX | 16#60 |
| PV_TL_Act.ST | 16#XX | 16#60 |
| PV_WL_Act.ST | 16#XX | 16#60 |
| PV_AL_Act.ST | 16#XX | 16#60 |
| PV_ToleHi.ST | 16#80 | 16#60 |
| PV_ToleLo.ST | 16#80 | 16#60 |
| ER.ST | 16#80 | 16#60 |
| ER_AH_Act.ST | 16#XX | 16#60 |
| ER_AL_Act.ST | 16#XX | 16#60 |

**Note**

The status of `RbkWH_Act` and `RbkWL_Act` will be 16#80 without Rbk.

### See also

Description of PIDStepL (Page 1141)
PIDStepL functions (Page 1147)
PIDStepL error handling (Page 1159)
PIDStepL messaging (Page 1160)
PIDStepL I/Os (Page 1163)
PIDStepL block diagram (Page 1179)

## 6.10.3 PIDStepL functions

### Functions of PIDStepL

The functions for this block are listed below.

### Generation of manipulated variables with position feedback (`WithRbk` = 1)

The manipulated variable `MV` and therefore the actuating signals `Open`, `Close` and `Stop` can be generated as follows:

| MV_ForOn | ManAct | MV_TrkOn | AdvCoAct AND NOT AdvCoModSP | MV = | Limit monitoring | State | Open, Close, Stop |
|---|---|---|---|---|---|---|---|
| 1 | - | - | - | `MV_Forced` | none | Forced tracking through constraint without limitation | Depending on `Rbk` and `MV`, the output signals `Open`, `Close` and `Stop` are generated using the algorithm of a positioner. |
| 0 | 1 | - | - | `Man` | `ManHiLim` `ManLoLim` | Manual mode, set by the operator | |
| 0 | 0 | 1 | - | `MV_Trk` | `MV_HiLim` `MV_LoLim` | Tracking with limitation | |
| 0 | 0 | 0 | 1 | `AdvCoMV` | `MV_HiLim` `MV_LoLim` | Higher-level program mode | |
| 0 | 0 | 0 | 0 | `P_Part` + `I_Part` + `D_Part` + `FFwd` | `MV_HiLim` `MV_LoLim` | Automatic mode (PID algorithm) | |

If the controller is in "out of service" mode, the output parameter `MV` is set to the last valid value in manual mode or the neutral position manipulated variable depending on the `Feature` Bit (Neutral position manipulated variable takes effect at startup (Page 176)). Refer to the Out of service (Page 77) section for more on this.

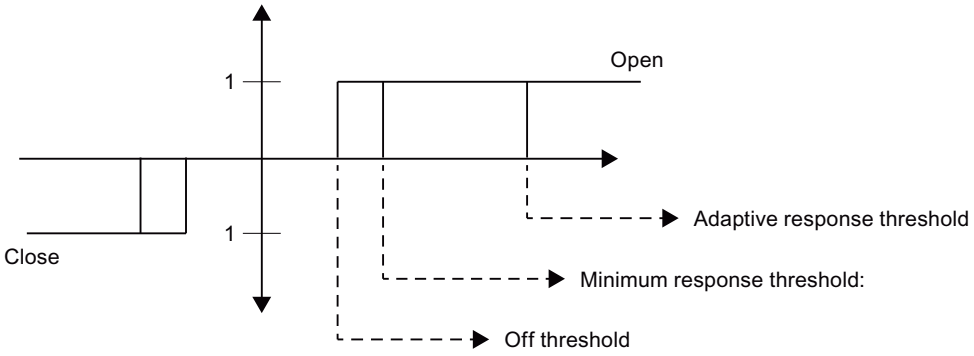
### Generation of actuating signal without position feedback (`WithRbk` = 0)

The manipulated variable `MV` can be generated as follows:

| ManAct | Open, Close, Stop | State |
|---|---|---|
| 1 | The output signals are generated using the input signals:<br>`Feature.Bit7` = 0<br>`ModLiOp` = 0: `OpenOp`, `CloseOp` or `StopOp`<br>`ModLiOp` = 1: `OpenLi`, `CloseLi` or `StopLi`<br>`Feature.Bit7` = 1<br>`CtrlLiOp` = 0: `OpenOp`, `CloseOp` or `StopOp`<br>`CtrlLiOp` = 1: `OpenLi`, `CloseLi` or `StopLi` | Manual mode |
| 0 | The output signals are generated using PD output variables `P_Part`, `D_Part` and `FFwd` | Automatic mode (PID algorithm) |

If the controller is in "out of service" mode, the output parameter `MV` is set to the last valid value in manual mode or the neutral position manipulated variable depending on the `Feature` Bit (Neutral position manipulated variable takes effect at startup (Page 176)). Refer to the Out of service (Page 77) section for more on this.

### Three-position element with hysteresis and pulse generation




The difference (`MV - Rbk`) for a step controller with position feedback forms the input for the three-position element. The difference for step controllers without position feedback is formed from the output of the PD block and an internal feedback, which is weighted by the common integration of the position signal Open/Close `MotorTime` and the I-component weighted by `TI` (see block diagram).

You can calculate the response threshold of the three position element as follows:

- Minimum response threshold: `100.0 * Max(PulseTime, SampleTime) / MotorTime`
- Off threshold: `0.5 * 110.0 / MotorTime * SampleTime;`

Using "`ThrAdaOn`" you can turn on an adaption of the response threshold to reduce the switching frequency. This only affects the closed control loop (automatic operation without tracking) and is limited to:

- Down: Minimum response threshold
- Up: 10

The currently effective response threshold can be read at the output "`ThesOn`".

The pulse generation after the three position element ensures that `PulseTime` and `BreakTime` are maintained during pulse generation.

### Tracking and limiting a manipulated variable

The block provides the standard function Tracking and limiting a manipulated variable (Page 222).

**Note**

This function is only available to you if position feedback is enabled for the controller (`WithRbk` = 1).

### Neutral position

The block provides the standard function Neutral position for motors, valves and controllers (Page 54).

### Group error

This block provides the standard function Outputting group errors (Page 130).

The following parameters are taken into consideration when forming group errors:

- `CSF`

### Outputting a signal for start readiness

This block provides the standard function Outputting a signal for start readiness (Page 59).

### "Actuator active" information

If the parameter `FbkClosed` = 0, this is recognized as "Actuator active". This status can be used to indicate a customized symbol in the process image, for example, and is saved in the status word (see Status word section in Description of PIDStepL (Page 1141)).

### Limit monitoring of position feedback

The block provides the standard function Limit monitoring of the feedback (Page 101).

**Note**

This function is only available to you if position feedback is enabled for the controller (`WithRbk` = 1).

### Group display SumMsgAct for limit monitoring, CSF and ExtMsgx

The block provides the standard function Group display for limit monitoring, CSF and ExtMsgx (Page 92).

### External/internal setpoint specification

The block provides the standard function Setpoint specification - internal/external (Page 136).

### Setpoint limiting for external setpoints

The block provides the standard function Setpoint limiting for external setpoints (Page 222).

### Gradient limit of the setpoint

The block provides the standard function Gradient limit of the setpoint (Page 132).

### Using setpoint ramp

The block provides the standard function Using setpoint ramp (Page 131).

### Tracking setpoint in manual mode

The block provides the standard function Tracking setpoint in manual mode (Page 222).

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Process value (`SimPV, SimPV_Li`)
- Position feedback (`SimRbk, SimRbkLi`)

### Bypass function

This block provides the standard function Bypassing signals (Page 115).

### Limit monitoring of the process value

The block provides the standard function Limit monitoring of the process value (Page 93).

### Providing limits at the output

For further connections to other blocks the following input parameters will also be displayed on the output:

`ER_AH_LimOut` := `ER_AH_Lim`

`ER_AL_LimOut` := `ER_AL_Lim`

### Error signal generation and dead band

The block provides the standard function Error signal generation and dead band (Page 217).

### Limit monitoring of error signal

The block provides the standard function Limit monitoring of setpoint, manipulated variable and control deviation (Page 102).

### Inverting control direction

The block provides the standard function Inverting control direction (Page 217).

### Physical standardization of setpoint, manipulated variable and process value

Controller gain `Gain` is entered either using a physical variable or as standardized value.

`Gain` as a physical variable:

The standardized variables retain their default values:

- `NormPV.High` = 100 and `NormPV.Low` = 0

The effective gain is:

`GainEff` = `Gain`

Entering a standardized `Gain` (dimensionless):

Change the standardized variables to the actual range of the process values and manipulated variables.

- Internal and external setpoints; the process value and corresponding parameters are entered according to the physical measuring range of the process value.
- The manual parameter, the tracking value of the manipulated variable, feedforward control and the corresponding parameters are entered as a percentage 0...100.

The effective gain is:

`GainEff` = 100.0 / (`NormPV.High` - `NormPV.Low`) · `Gain`

## Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 240).

## PID algorithm

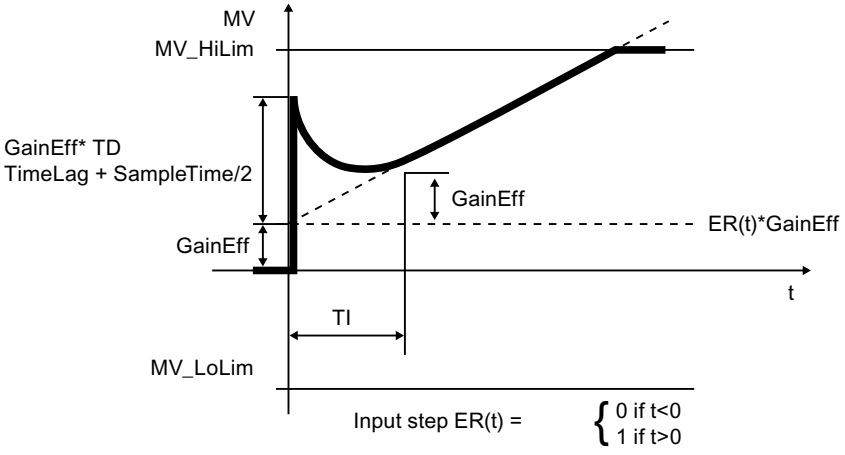
The manipulated variable is generated in automatic mode according to the following algorithm:

`MV` = `GainEff` · (1 + 1 / (`TI` · s) + (`TD` · s) / (1 + `TD` / `DiffGain` · s)) · `ER`

Where:

s = Complex number

The following step response occurs:




**Note**

The formula describes a standard application where P, I and D components are activated and the P and D components are not in the feedback circuit (`PropSel` = 1, `TI` <> 0, `DiffToFbk` = 0 `and IntSel` = 1, `DiffToFbk` = 0 and `PropFacSP` = 1).

The D component delay is derived from TD / DiffGain.

- The P component is at the P_Part I/O and can be deactivated using PropSel = 0 .
- The I component is displayed at the I_Part I/O and can be deactivated using TI = 0 or IntSel = 0. In deactivated state, I_Part is specified by MV_Offset and added to the manipulated variable. Make a selection for this value so that the remaining control deviation equals zero at the control loop's typical operating point, at least. IntSel is used for temporary deactivation of the I component. The I component is not reactivated until TI <> 0 and IntSel = 1. After the I component is activated, the integrator continues working starting from MV_Offset.
- The D component is displayed at the D_Part I/O and can be deactivated using TD = 0 or DiffSel = 0.

### Structure segmentation at controllers

The block provides the standard function Structure segmentation at controllers (Page 224).

### Anti-windup

The controller has an anti-windup function. The I component is frozen after the manipulated variable has reached limits (MV_HiLim or MV_LoLim).

### Feedforwarding and limiting disturbance variables

The block provides the standard function Feedforwarding and limiting disturbance variables (Page 223).

### Control loop is open (CascaCut)

The CascaCut output indicates whether the control loop within this block is open. In the open control loop, the master controller must be set to tracking mode to avoid until windup behavior. If you use this block in a cascade structure, then connect the output CascaCut to the input MV_TrkOn of the master controller.

CascaCut is active in the following cases:

- Block is not in automatic mode
- Setpoint is not in external mode
- Force or Tracking function is active

### Forming the signal status for blocks

The block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

- Signal status for the process value `PV_Out`:
  The signal status of the output parameter `PV_Out` always corresponds to the signal status of input parameter `PV` or, if the block is in simulation mode, 16#60.
- Signal status for the setpoint value `SP`:
  The signal status of the `SP` output parameter is always equivalent to the signal status of input parameter `SP_Ext` or `SP_Int`, depending on how the setpoint is specified. If the internal setpoint `SP_Int` is used, the signal status is always output as 16#80.
- Signal status of the error signal `ER`:
  The signal status of output parameter `ER` is obtained from the worst signal status of the two output parameters `PV_Out` and `SP` and is output.
  The signal status `16#60` (external simulation) is suppressed because the block acts as a sink with external simulation.
- Signal status for position feedback `RbkOut`:
  The signal status of `RbkOut` always corresponds to the signal status of input parameter `Rbk` or, if the block is in simulation mode, `16#60`. `RbkOut` is always `16#80` for step controllers without position feedback.
- Signal status for the manipulated variable `MV`:
  The signal status of output parameter `MV` is obtained in "automatic mode" from the worst signal status of the following parameters and is output:
  `ER.STFFwd.STFbkOpened.STFbkClosed.STRbkOut.ST`The signal status `16#60` (external simulation) is suppressed because the block acts as a sink with external simulation.
  In "manual mode" and for step controllers without position feedback, the signal status is always output as good.

- Signal status for `Open`, `Close`, `Stop`:
  The signal status is `16#60` when simulation is activated; otherwise, it is always `16#80`.
- Worst signal status:
  The worst signal status `ST_Worst` for the block is formed from the following parameters:
  - `SP.ST`;
  - `PV_Out.ST`;
  - `FFwd.ST`;
  - `FbkOpened.ST`;
  - FbkClosed.ST;
  - `RbkOut.ST`;
  - `OpenChnST.ST`;
  - `CloseChnST.ST`;
  - `StopChnST.ST`;

**Note**

The `RbkOut.ST` parameter is always 16#80for a step controller without position feedback (`WithFbk` = 0) .

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 4 | Setting switch or button mode (Page 177) |
| 5 | Changes signal status of outputs in OOS (Page 208) |
| 6 | Ramp rate calculation (Page 191) |
| 7 | Use connectable inputs OpenLi, StopLi, CloseLi with CtrlLiOp (Page 212) |
| 9 | Substitution value is active if the block is in bypass (Page 199) |
| 11 | Gradient limitation with time duration (Page 197) |
| 15 | Neutral position manipulated variable takes effect with "out of service" operating mode (Page 177) |
| 16 | Neutral position manipulated variable takes effect at startup (Page 176) |
| 18 | Disabling bumpless switchover to automatic mode for controllers (Page 183) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 23 | SP following PV in open loop has no priority over SP_Ext and SP limits (Page 193) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |

| Bit | Function |
|---|---|
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 28 | Disabling operating points (Page 156) |
| 29 | Signaling limit violation (Page 181) |
| 31 | Limits disabled in faceplate (Page 206) |

### Configurable reactions using the `Feature2` parameter

| Bit | Function |
|---|---|
| 11 | Suppress MsgLock and “Out of service” mode for a connected message block (Page 202) |
| 12 | Disable limit act output when limit value message is disabled (Page 209) |

### Operator permissions

The block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" `AutModOp` |
| 1 | 1 = Operator can switch to "manual mode" `ManModOp` |
| 2 | 1 = Operator can switch to "Out of service" mode `OosOp` |
| 3 | 1 = Operator can switch to "program mode" `AdvCoEn` |
| 4 | 1 = Operator can switch the setpoint to "external" `SP_ExtOp` |
| 5 | 1 = Operator can switch the setpoint to "internal" `SP_IntOp` |
| 6 | 1 = Operator can change the internal setpoint `SP_Int` |
| 7 | 1 = Operator can change the manual parameter `Man` (`WithRbk` = 1) |
| 8 | 1 = Operator can change operation high limit of the setpoint `SP_InHiLim` |
| 9 | 1 = Operator can change operation low limit of the setpoint `SP_InLoLim` |
| 10 | 1 = Operator can change the operation high limit of the manipulated variable `ManHiLim` |
| 11 | 1 = Operator can change the operation low limit of the manipulated variable `ManLoLim` |
| 12 | 1 = Operator can enable the setpoint's gradient limitation function `SP_RateOn` |
| 13 | 1 = Operator can change the setpoint's high limit for the ramp `SP_UpRaLim` |
| 14 | 1 = Operator can change the setpoint's low limit for the ramp `SP_DnRaLim` |
| 15 | 1 = Operator can switch between the time value or the value for the ramp `SP_RmpModTime` |
| 16 | 1 = Operator can change the ramp time `SP_RmpTime` |
| 17 | 1 = Operator can change the target setpoint `SP_RmpTarget` for the setpoint ramp |
| 18 | 1 = Operator can enable the setpoint ramp function `SP_RmpOn` |
| 19 | 1 = Operator can permit the PID optimization function `OptimEn` |
| 20 | 1 = Operator can enable the track setpoint in "manual mode" function `SP_TrkPV` |
| 21 | 1 = Operator can enable the bumpless switchover from external to internal `SP_TrkExt` |
| 22 | 1 = Operator can change the gain parameter `Gain` |
| 23 | 1 = Operator can change the integral time parameter `TI` |
| 24 | 1 = Operator can change the derivative time parameter `TD` |

| Bit | Function |
|---|---|
| 25 | 1 = Operator can change the derivative gain parameter `DiffGain` |
| 26 | 1 = Operator can change the dead band parameter `DeadBand` |
| 27 | 1 = Operator can activate bypass functionality |
| 28 | 1 = Operator can change the integral time parameter `MotorTime` |
| 29 | 1 = Operator can change the integral time parameter `PulseTime` |
| 30 | 1 = Operator can change the integral time parameter `BreakTime` |
| 31 | 1 = Operator can deactivate bypass functionality |

The block has the following permissions for the `OS1Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can change the limit (process value) `PV_AH_Lim` for the high alarm |
| 1 | 1 = Operator can change the limit (process value) `PV_WH_Lim` for the high warning |
| 2 | 1 = Operator can change the limit (process value) `PV_TH_Lim` for the high tolerance |
| 3 | 1 = Operator can change the hysteresis (process value) `PV_Hyst` |
| 4 | 1 = Operator can change the limit (process value) `PV_TL_Lim` for the low tolerance |
| 5 | 1 = Operator can change the limit (process value) `PV_WL_Lim` for the low warning |
| 6 | 1 = Operator can change the limit (process value) `PV_AL_Lim` for the low alarm |
| 7 | 1 = Operator can change the limit (error signal) `ER_AH_Lim` for the high alarm |
| 8 | 1 = Operator can change the hysteresis (error signal) `ER_Hyst` |
| 9 | 1 = Operator can change the limit (error signal) `ER_AL_Lim` for the low alarm |
| 10 | 1 = Operator can change the limit (position feedback) `RbkWH_Lim` for the high warning |
| 11 | 1 = Operator can change the hysteresis (position feedback) `RbkHyst` |
| 12 | 1 = Operator can change the limit (position feedback) `RbkWL_Lim` for the low warning |
| 13 | 1 = Operator can open the valve: `OpenOp` (`WithRbk` = 0) |
| 14 | 1 = Operator can close the valve: `CloseOp` (`WithRbk` = 0) |
| 15 | 1 = Operator can stop the valve: `StopOp` (`WithRbk` = 0) |
| 16 | 1 = Operator can activate the Simulation function `SimOn` |
| 17 | 1 = Operator can activate the Release for maintenance function `MS_RelOp` |
| 18 | 1 = Operator can activate / deactivate messages via `PV_AH_MsgEn` |
| 19 | 1 = Operator can activate / deactivate messages via `PV_WH_MsgEn` |
| 20 | 1 = Operator can activate / deactivate messages via `PV_TH_MsgEn` |
| 21 | 1 = Operator can activate / deactivate messages via `PV_TL_MsgEn` |
| 22 | 1 = Operator can activate / deactivate messages via `PV_WL_MsgEn` |
| 23 | 1 = Operator can activate / deactivate messages via `PV_AL_MsgEn` |
| 24 | 1 = Operator can activate / deactivate messages via `ER_AH_MsgEn` |
| 25 | 1 = Operator can activate / deactivate messages via `ER_AL_MsgEn` |
| 26 | 1 = Operator can activate / deactivate messages via `RbkWH_MsgEn` |
| 27 | 1 = Operator can activate / deactivate messages via `RbkWL_MsgEn` |
| 28 | 1 = Operator can change the simulation value `SimPV` |
| 29 | 1 = Operator can change the simulation value `SimRbk` |
| 18 - 29 | Not used |

| Bit | Function |
|---|---|
| 30 | 1 = Operator can activate the derivative action to the feedback path `DiffToFbk` |
| 31 | 1 = Operator can change the proportional action to the feedback path `PropFacSP` |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Release for maintenance

The block provides the standard function Release for maintenance (Page 70).

### Generating instance-specific messages

The block provides the standard function Generating instance-specific messages (Page 230) without the time stamp function in the I/O.

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Specifying the display area for process and setpoint values as well as operations

This block provides the standard function Display and operator input area for process values and setpoints (Page 233).

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Button labels

This block provides the standard function Labeling of buttons and text (Page 235)

Instance-specific text can be configured for the following parameters:

- `OpenOp`
- `StopOp`
- `CloseOp`

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### See also

PIDStepL messaging (Page 1160)

PIDStepL I/Os (Page 1163)

PIDStepL error handling (Page 1159)

PIDStepL modes (Page 1146)

PIDStepL block diagram (Page 1179)

Evaluation of the signal status of the interlock signals (Page 153)

## 6.10.4 PIDStepL error handling

### Error handling of PIDStepL

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Control system fault (CSF)

### Overview of error numbers

The `ErrorNum` output parameter can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. |
| 30 | The value of `PV` can no longer be displayed in the REAL number field. |
| 31 | The value of `SP_Ext` can no longer be displayed in the REAL number field. |
| 32 | The value of `FFwd` can no longer be displayed in the REAL number field. |
| 33 | The value of `MV_Trk` can no longer be displayed in the REAL number field. |
| 34 | The value of `MV_Forced` can no longer be displayed in the REAL number field. |
| 35 | The value of `Rbk` can no longer be displayed in the REAL number field. |
| 36 | The value of `MV` can no longer be displayed in the REAL number field. |
| 50 | The controller cannot be switched to program mode, because program mode with default setpoint (`AdvCoModSP` = 0) is not possible for step controllers without position feedback (`WithRbk` = 0). |

| Error number | Meaning of the error number |
|---|---|
| 51 | AutModLi = 1 and ManModLi = 1<br>SP_LiOp = 1 and SP_IntLi = 1 and SP_ExtLi = 1<br>OpenLi = 1 and StopLi = 1<br>CloseLi = 1 and StopLi = 1<br>OpenLi = 1 and CloseLi = 1 |
| 59 | = 1 "Gain is negative" |
| 60 | \|TI\| < SampleTime / 2 |
| 61 | \|TD\| < SampleTime |
| 62 | DiffGain < 1 or DiffGain > 10 |
| 63 | TD / DiffGain < SampleTime / 2 |
| 64 | PropFacSP < 0 or PropFacSP > 1 |
| 66 | NormPV_High = NormPV_Low |
| 67 | MotorTime < SampleTime |
| 68 | PulseTime < SampleTime |
| 69 | BreakTime < SampleTime |

### Control system fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 128) section for more on this.

### See also

PIDStepL functions (Page 1147)

Description of PIDStepL (Page 1141)

PIDStepL modes (Page 1146)

PIDStepL messaging (Page 1160)

PIDStepL I/Os (Page 1163)

PIDStepL block diagram (Page 1179)

Setting switch or button mode (Page 177)

## 6.10.5 PIDStepL messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Process messages
- Instance-specific messages

### Process control fault

The following control system error messages can be output:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId2 | SIG 6 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If it changes to CSF = 1, a process control fault is triggered (MsgEvId2, SIG 6).

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 1 | Alarm - high | $$BlockComment$$<br>PV - high alarm limit violated |
| | SIG 2 | Warning - high | $$BlockComment$$<br>PV - high warning limit violated |
| | SIG 3 | Tolerance - high | $$BlockComment$$<br>PV - high tolerance limit violated |
| | SIG 4 | Tolerance - low | $$BlockComment$$<br>PV - low tolerance limit violated |
| | SIG 5 | Warning - low | $$BlockComment$$<br>PV - low warning limit violated |
| | SIG 6 | Alarm - low | $$BlockComment$$<br>PV - low alarm limit violated |
| | SIG 7 | Alarm - high | $$BlockComment$$<br>ER - high alarm limit violated |
| | SIG 8 | Alarm - low | $$BlockComment$$<br>ER - low alarm limit violated |
| MsgEvId2 | SIG 7 | Warning - high | $$BlockComment$$<br>Rbk - high warning limit violated |
| | SIG 8 | Warning - low | $$BlockComment$$<br>Rbk - low warning limit violated |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Instance-specific messages

You can use up to four instance-specific messages with this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId2 | SIG 1 | AS process control message - fault | $$BlockComment$$<br>External message 1<br>Status 16#@5%x@ |
| | SIG 2 | AS process control message - fault | $$BlockComment$$<br>External message 2<br>Status 16#@6%x@ |
| | SIG 3 | AS process control message - fault | $$BlockComment$$<br>External message 3<br>Status 16#@7%x@ |
| | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External message 4<br>Status 16#@8%x@ |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance **MsgEvId1**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |
| 4 | Process value PV_Out |
| 5 | Control deviation ER |
| 6 | ExtVa106 |
| 7 | ExtVa107 |
| 8 | Not allocated |
| 9 | Not allocated |
| 10 | Not allocated |

The associated values 6 ... 7 are allocated to the parameters ExtVa106 ... ExtVa107 and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### Associated values for message instance **MsgEvId2**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |

| Associated value | Block parameters |
|---|---|
| 4 | Position feedback RbkOut |
| 5 | Signal status ExtMsg1 |
| 6 | Signal status ExtMsg2 |
| 7 | Signal status ExtMsg3 |
| 8 | Signal status ExtMsg4 |
| 9 | ExtVa209 |
| 10 | ExtVa210 |

The associated values 9 ... 10 are allocated to the parameters ExtVa209 ... ExtVa210 and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of PIDStepL (Page 1141)

PIDStepL functions (Page 1147)

PIDStepL I/Os (Page 1163)

PIDStepL modes (Page 1146)

PIDStepL error handling (Page 1159)

PIDStepL block diagram (Page 1179)

## 6.10.6 PIDStepL I/Os

I/Os of PIDStepL

For the parameters that are not linkable, refer:Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AdvCoEn | 1 = Enable "program mode" via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AdvCoOn* | 1 = Enable "program mode" via faceplate.<br>The parameter is available on the WinCC OPC server to be read by external APC programs. | BOOL | 0 |
| AdvCoModSP | Type of "program mode":<br>1 = Setpoint specification<br>0 = Manipulated variable specification | BOOL | 1 |
| AdvCoMstrOn | Activate (0-1) or deactivate (1-0) "program mode" via edge transition | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| AdvCoMV | Specified value from the external program.<br>The parameter is available on the WinCC OPC server to be read by external APC programs. | REAL | 0.0 |
| AutModLi* | 1 = "Automatic mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutModOp* | 1 = "Automatic mode" via operator (controlled by ModLiOp = 0) | BOOL | 0 |
| BatchEn | 1 = Enable allocation for batch control | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of block-type<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkTypeData* | This input can not be edit or linked and is for internal usage only! | | |
| BreakTime | Minimum break duration [s] | REAL | 1.0 |
| BypLiOp | 1 = Bypass commands via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypLock | 1 = Bypass activation or deactivation is locked for operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypProt | Bypass Protection in Sim Modes | BOOL | 0 |
| BypPV | Substitution value if block is in bypass | REAL | 0.0 |
| BypPVLi | 1 = Select bypass PV (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypPVOp | 1 = Select bypass PV (via operator) | BOOL | 0 |
| CloseChnST | Output channel state of Close<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| CloseLi* | 1 = Close via interconnection or CFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CloseOp* | 1 = Close via operator | BOOL | 0 |
| CPI_In | Input for control performance index, which is calculated by the assigned ConPerMon block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#78 |
| CSF | 1 = External error (control system fault)<br>Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DeadBand | Width of dead band | REAL | 0.0 |
| DiffGain | Gain of differentiator [1..10] DiffGain = TD / (delay time of D component) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 5.0<br>• 16#FF |
| DiffSel | 1 = D component activated | BOOL | 1 |
| DiffToFbk | 1 = D component is placed in the feedback | BOOL | 0 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| ER_A_DC* | Delay for incoming alarms during error signal monitoring | REAL | 0.0 |
| ER_A_DG* | Delay for outgoing alarms during error signal monitoring | REAL | 0.0 |
| ER_AH_En | 1 = Activate alarm (high) for error signal monitoring | BOOL | 1 |
| ER_AH_Lim | Alarm limit (high) for error signal monitoring | REAL | 100.0 |
| ER_AH_MsgEn | 1 = Activate messages for alarm (high) for error signal monitoring | BOOL | 1 |
| ER_AL_En | 1 = Activate alarm (low) for error signal monitoring | BOOL | 1 |
| ER_AL_Lim | Alarm limit (low) for error signal monitoring | REAL | -100.0 |
| ER_AL_MsgEn | 1 = Activate messages for alarm (low) for error signal monitoring | BOOL | 1 |
| ER_Hyst | Alarm hysteresis for error signal | REAL | 1.0 |
| EventTsIn | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ExtMsg1 | 1 = Binary input for freely selectable message 1 is used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | 1 = Binary input for freely selectable message 2 is used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg3 | 1 = Binary input for freely selectable message 3 is used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg4 | 1 = Binary input for freely selectable message 4 is used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| ExtVa209 | Associated value 9 for messages (MsgEvID2) | ANY | |
| ExtVa210 | Associated value 10 for messages (MsgEvID2) | ANY | |
| FbkClosed | Low limit stop signal of position feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>0<br>16#80 |
| FbkOpened | High limit stop signal of position feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Feature | I/O for additional functions (Page 1147) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| FFwd* | Input for additive disturbance variable activation | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| FFwdHiLim | Limit (high) for additive disturbance variable activation | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| FFwdLoLim | Limit (low) for additive disturbance variable activation | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• -100.0<br>• 16#80 |
| Gain | Proportional gain<br>Gain.ST = 16#FF: Enabled in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#FF |
| IntHoldNeg | 1 = Integrator cannot run in negative direction | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| IntHoldPos | 1 = Integrator cannot run in positive direction | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Intlock | Reserved | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| Intl_En | Reserved | BOOL | 1 |
| IntSel | 1 = I component activated | BOOL | 1 |
| Man* | Manual specification for the manipulated variable | REAL | 0.0 |
| ManHiLim | Limit (high) for manual parameter Man | REAL | 100.0 |
| ManLoLim | Limit (low) for manual parameter Man | REAL | 0.0 |
| ManModLi* | 1 = "Manual mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManModOp* | 1 = "Manual mode" via OS operator (controlled by ModLiOp = 0) | BOOL | 1 |
| ModLiOp | Operating mode switchover between:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MotorTime | Motor actuating time [s] | REAL | 30.0 |
| MS_RelOp* | 1 = Release for maintenance by OS operator | BOOL | 0 |
| MsgEvID1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgEvID2 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_Forced* | Forced manipulated variable that is not limited and assumes top priority | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_ForOn | 1 = Output forced manipulated variable MV_Forced unlimited at output MV | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_HiLim | Limit (high) for manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| MV_LoLim | Limit (low) for manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_Offset | Manipulated variable for ER = 0, operating point for controller with deactivated I component | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MV_Trk* | Tracking value for the manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_TrkOn | 1 = Tracking of manipulated variable MV | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NegGain | 0 = Effective proportional gain GainEff is positive<br>1 = Effective proportional gain GainEff is negative | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NormPV | Process value range (PV) for standardizing the proportional gain (GAIN) | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpenChnST | Output channel state of Open<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| OpenLi* | 1 = Open via interconnection or CFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpenOp* | 1 = Open via operator | BOOL | 0 |
| OptimEn* | 1 = Enable optimization of PID parameters by PID tuner | BOOL | 0 |
| OptimOcc* | 1 = Optimization running | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 1147) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| OS1Perm | I/O for operator permissions (Page 1147) | STRUCT<br>• Bit 0: BOOL<br>• Bit 18: BOOL<br>• Bit 19: BOOL<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1<br>• 1 |
| PropFacSP | Applying the P component to the feedback [0..1]. 0 = P component fully in feedback | REAL | 1.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PropSel | 1 = Activate P component | BOOL | 1 |
| PulseTime | Minimum pulse duration [s] | REAL | 1.0 |
| PV* | Process value (controlled variable) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_Unit | Unit of measure for process value | INT | 1001 |
| PV_A_DC* | Delay time for incoming PV alarms [s] | REAL | 0.0 |
| PV_A_DG* | Delay time for outgoing PV alarms [s] | REAL | 0.0 |
| PV_AH_En | 1 = Enable PV alarm limit (high) | BOOL | 1 |
| PV_AH_Lim | Limit PV alarm (high) | REAL | 95.0 |
| PV_AH_MsgEn | 1 = Enable PV alarm (high) message | BOOL | 1 |
| PV_AL_En | 1 = Enable PV alarm limit (low) | BOOL | 1 |
| PV_AL_Lim | PV alarm limit (low) | REAL | 5.0 |
| PV_AL_MsgEn | 1 = Enable PV alarm (low) message | BOOL | 1 |
| PV_Hyst | Hysteresis for PV alarm, warning and tolerance limits | REAL | 1.0 |
| PV_OpScale | Limit for scale in PV bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| PV_T_DC* | Delay time for incoming PV tolerance messages [s] | REAL | 0.0 |
| PV_T_DG* | Delay time for outgoing PV tolerance messages [s] | REAL | 0.0 |
| PV_TH_En | 1 = Enable PV tolerance limit (high) | BOOL | 0 |
| PV_TH_Lim | Limit PV tolerance message (high) | REAL | 85.0 |
| PV_TH_MsgEn | 1 = Enable message for PV tolerance message (high) | BOOL | 1 |
| PV_TL_En | 1 = Enable PV tolerance limit (low) | BOOL | 0 |
| PV_TL_Lim | Limit PV tolerance message (low) | REAL | 15.0 |
| PV_TL_MsgEn | 1 = Activate message for PV tolerance message (low) | BOOL | 1 |
| PV_Unit | Unit of measure for process value | INT | 1001; °C |
| PV_W_DC* | Delay time for incoming PV warnings [s] | REAL | 0.0 |
| PV_W_DG* | Delay time for outgoing PV warnings [s] | REAL | 0.0 |
| PV_WH_En | 1 = Enable PV warning limit (high) | BOOL | 1 |
| PV_WH_Lim | Limit PV warning (high) | REAL | 90.0 |
| PV_WH_MsgEn | 1 = Enable PV warning (high) message | BOOL | 1 |
| PV_WL_En | 1 = Enable PV warning limit (low) | BOOL | 1 |
| PV_WL_Lim | Limit PV warning (low) | REAL | 10.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_WL_MsgEn | 1 = Enable PV warning (low) message | BOOL | 1 |
| Rbk* | Position feedback | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| RbkHyst | Alarm hysteresis for position feedback | REAL | 1.0 |
| RbkWH_En | 1 = Enable warning (high) for position feedback | BOOL | 1 |
| RbkWH_Lim | Limit for position feedback of warning (high) | REAL | 100.0 |
| RbkWH_MsgEn | 1 = Enable messages for warning (high) for position feedback | BOOL | 1 |
| RbkWL_En | 1 = Enable warning (low) for position feedback | BOOL | 1 |
| RbkWL_Lim | Limit for position feedback of warning (low) | REAL | 0.0 |
| RbkWL_MsgEn | 1 = Enable messages for warning (low) for position feedback | BOOL | 1 |
| RefStdDevIn | Reference value of PV standard deviation (sigma) in defined "good" state of control loop | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#78 |
| RstBypLi | 1 = Reset bypass PV (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstBypOp | 1 = Reset bypass PV (via operator) | BOOL | 0 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| S_RbkOnPIDTun | Simulation of position feedback on;<br>For PCS 7 PID tuner only | BOOL | 0 |
| S_RbkPIDTun* | Simulated position feedback | REAL | 50.0 |
| SafePos | Neutral position (Page 54) for step controller actuating signals:<br>0 = Close<br>1 = Open<br>2 = Stop | INT | 0 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | 1 = Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SelFp2 | 1 = Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | - |
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SimPV* | Process value used for SimOn = 1 | REAL | 0.0 |
| SimPV_Li | Process value that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SimRbk* | Position feedback used for SimOn = 1 | REAL | 0.0 |
| SimRbkLi | Position feedback used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_DnRaLim* | Limit (low) for the gradient of the setpoint [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 100.0 |
| SP_ExHiLim | Limit (high) for external setpoint | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| SP_ExLoLim | Limit (low) for external setpoint | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_Ext* | External setpoint of - (to interconnection) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_ExtLi* | 1 = Select external setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtOp* | 1 = Select external setpoint (via operator) | BOOL | 0 |
| SP_InHiLim | Limit (high) of internal setpoint | REAL | 100.0 |
| SP_InLoLim | Limit (low) of internal setpoint | REAL | 0.0 |
| SP_Int* | Internal setpoint for operation | REAL | 0.0 |
| SP_IntLi* | 1 = Select internal setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_IntOp* | 1 = Select internal setpoint (via operator) | BOOL | 0 |
| SP_LiDnRaLim | Linkable limit (low) for the gradient of the setpoint [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_LiOp | Select setpoint source (internal/external):<br>1 = Via interconnection<br>0 = Via operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP_LiRmpTime | Linkable time [Time unit] for gradient limitation in ramp mode duration<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_LiUpRaLim | Linkable gradient limit (high) for the setpoint [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_RateOn* | 1 = Activate limitation of setpoint gradients | BOOL | 0 |
| SP_RmpModTime | 1 = Use time values (SP_RmpTime, SP_LiRmpTime)<br>0 = Use gradient limits (SP_DnRaLim, SP_LiDnRaLim, SP_UpRaLim, SP_LiUpRaLim) | BOOL | 0 |
| SP_RmpOn* | 1 = Activate setpoint ramp to target setpoint SP_RmpTarget | BOOL | 0 |
| SP_RmpTarget* | Target setpoint for setpoint ramp | REAL | 0.0 |
| SP_RmpTime* | Time [Time unit] for setpoint ramp from current SP up to SP_RmpTarget or time [Time unit] for gradient limitation in ramp mode duration<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_TrkExt | 1 = Bumpless switchover from external to internal setpoint active | BOOL | 1 |
| SP_TrkPV | 1 = Setpoint follows PV in "manual mode" and with tracking | BOOL | 0 |
| SP_UpRaLim* | Gradient limit (high) for the setpoint [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 100.0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| StopChnST | Output channel state of Stop<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| StopLi* | 1 = Stop via interconnection or CFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopOp* | 1 = Stop via operator | BOOL | 0 |
| TD | Derivative time in [s]<br>TD.ST = 16#FF: Enabled in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| ThrAdaOn | Adaptation of threshold<br>0 = Hold constant | BOOL | 1 |
| TI | Integral time [s]<br>TI.ST = 16#FF: Enabled in faceplate | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| TimeFactor | Time unit:<br>0 = Seconds<br>1 = Minutes<br>2 = Hours | INT | 0 |
| UA1unit | Unit of measure for analog auxiliary value 1 | INT | 0 |
| UA2unit | Unit of measure for analog auxiliary value 2 | INT | 0 |
| UserAna1 | Analog auxiliary value 1 | STRUCT<br>Value: REAL<br>ST: BYTE | <br>0.0<br>16#FF |
| UserAna2 | Analog auxiliary value 2 | STRUCT<br>Value: REAL<br>ST: BYTE | <br>0.0<br>16#FF |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus 2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |
| WithRbk | 1 = Feedback value available for the manipulated variable | BOOL | 0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AdvCoAct | 1 = "Program mode" enabled.<br>The parameter is available on the WinCC OPC server to be read by external APC programs. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AdvCoRdy | 1 = "Program mode" available | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutAct | 1 = "Automatic mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypassAct | 1 = Bypass is activated in this block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CascaCut | Cascade connection:<br>1 = Control chain from primary to secondary controller is interrupted | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Close | Control output:<br>1 = Closed is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| D_Part | D component of PID algorithm | REAL | 0.0 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ER | Error signal | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ER_AH_Act | 1 = Alarm limit (high) for control deviation violated.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| ER_AL_Act | 1 = Alarm limit (low) for control deviation violated.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| ER_AH_LimOut | ER_AH_Lim as output | REAL | 0.0 |
| ER_AL_LimOut | ER_AL_Lim as output | REAL | 0.0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see PIDStepL error handling (Page 1159) | INT | -1 |
| FFwdHiAct | 1 = Limit (high) for additive disturbance variable activation violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FFwdLoAct | 1 = Limit (low) for additive disturbance variable activation violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| GainEff | Effective proportional gain depends on NegGain, Gain, NormPV, and NormMV | REAL | 1.0 |
| GrpErr | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IntlockOut | Reserved | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| I_Part | I component of PID algorithm | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| LoopClosed | 1 = Control loop closed<br>0 = Control loop open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManAct | 1 = "Manual mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| ManHiOut | Limit (high) for "manual mode", corresponds to input parameter ManHiLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| ManLoOut | Limit (low) for "manual mode", corresponds to input parameter ManLoLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MS_Release | Release for maintenance: 1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message acknowledgement status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |
| MsgAckn2 | Message acknowledgement status 2 (output ACK_STATE of second ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | 1 = Alarm error 1 (output ERROR of the first ALARM_8P) | BOOL | 0 |
| MsgErr2 | 1 = Alarm error 2 (output ERROR of the second ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| MsgStat2 | Message status 2 (output STATUS of second ALARM_8P) | WORD | 16#0000 |
| MV | Manipulated variable | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_HiAct | 1 = Limit (high) of manipulated variable violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_LoAct | 1 = Limit (low) of manipulated variable violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Block is "out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Open | Control output:<br>1 = Open is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1Perm | DWORD | 16#FFFFFFFF |
| OS1PermLog | Display of OS1Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| P_Part | P component of PID algorithm | REAL | 0.0 |
| PV_AH_Act | 1 = PV alarm (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_AL_Act | 1 = PV alarm (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_Out | Output for process value | STRUCT<br>• Value: REAL<br>• ST: BYTE<br>• Bit 1: BOOL of internal fill byte (Bypass information bit) | -<br>• 0.0<br>• 16#80<br>• 0 |
| PV_OpScaleOut | PV_OpScale as output | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| PV_TH_Act | 1 = PV tolerance message (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_TL_Act | 1 = PV tolerance message (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_ToleHi | Limit (high) of 3-sigma band around the set-point is calculated when a ConPerMon block is connected | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_ToleLo | Limit (low) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_UnitOut | Unit of measure for process value, for interconnecting to the PV_Unit input parameter of the ConPerMon block | INT | 0 |
| PV_WH_Act | 1 = PV warning (high) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| PV_WL_Act | 1 = PV warning (low) active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| RbkOut | Output for position feedback | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RbkWH_Act | 1 = Warning (high) for position feedback active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RbkWL_Act | 1 = Warning (low) for position feedback active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToStart | 1 = Active start readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP | Setpoint used by controller | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_DnRaAct | 1 = Negative gradient limiting of setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP_ExHiAct | 1 = Limit (high) for external setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExLoAct | 1 = Limit (low) for external setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtAct | 1 = External setpoint is active<br>0 = Internal setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtOut | External setpoint, corresponds to input parameter SP_Ext | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_InHiOut | Limit (high) for SP_Int corresponds to input parameter SP_InHiLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| SP_InLoOut | Limit (low) for SP_Int corresponds to input parameter SP_InLoLim | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_RateTarget | Target setpoint for the gradient limitation | REAL | 0.0 |
| SP_UpRaAct | Positive gradient limiting of setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_RemRT | Remaining ramp time of the setpoint | REAL | 0.0 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 1141) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1141) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1141) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 1141) | DWORD | 16#00000000 |
| Stop | Control output:<br>1 =Stopped is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SumMsgAct | 1 = Active hardware interrupt | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Threson | Adaptive threshold [%] | REAL | 0.0 |
| UA1unitOut | UA1 as output | INT | 0 |
| UA2unitOut | UA2 as output | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| UserAna1Out | UserAna1 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| UserAna2Out | UserAna2 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |

### See also

PIDStepL messaging (Page 1160)

PIDStepL modes (Page 1146)

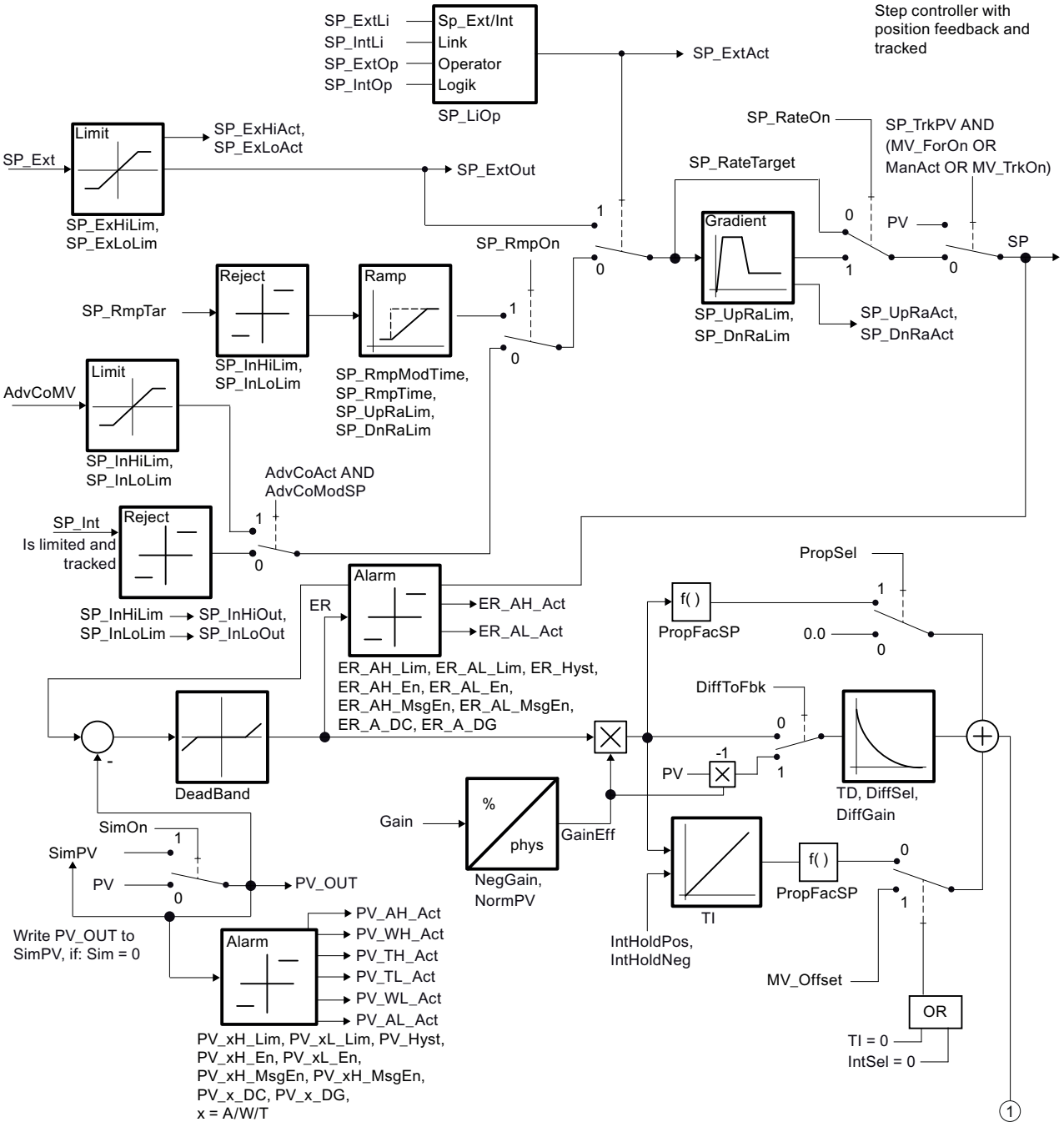
PIDStepL block diagram (Page 1179)

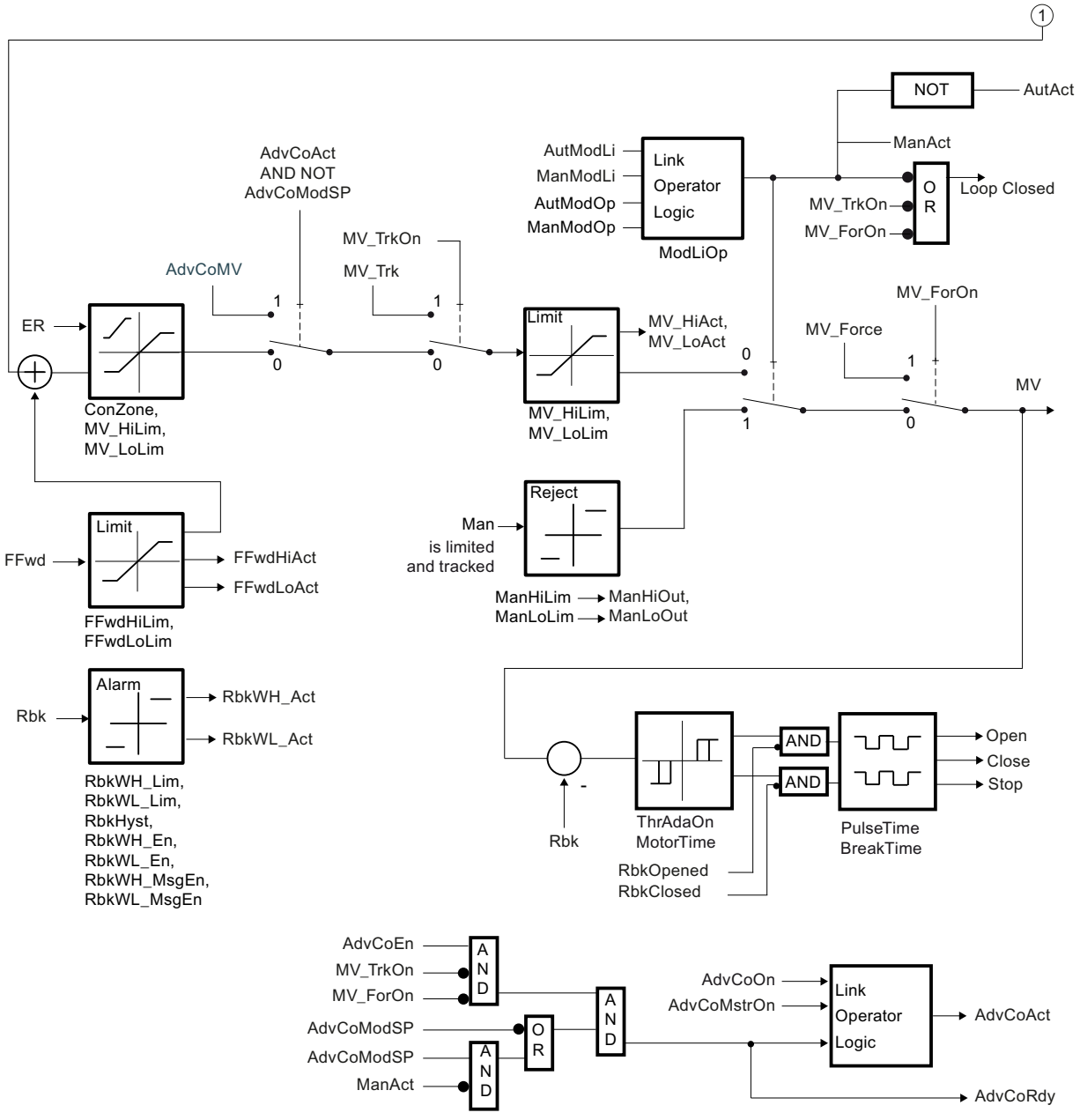
## 6.10.7 PIDStepL block diagram

### PIDStepL block diagram

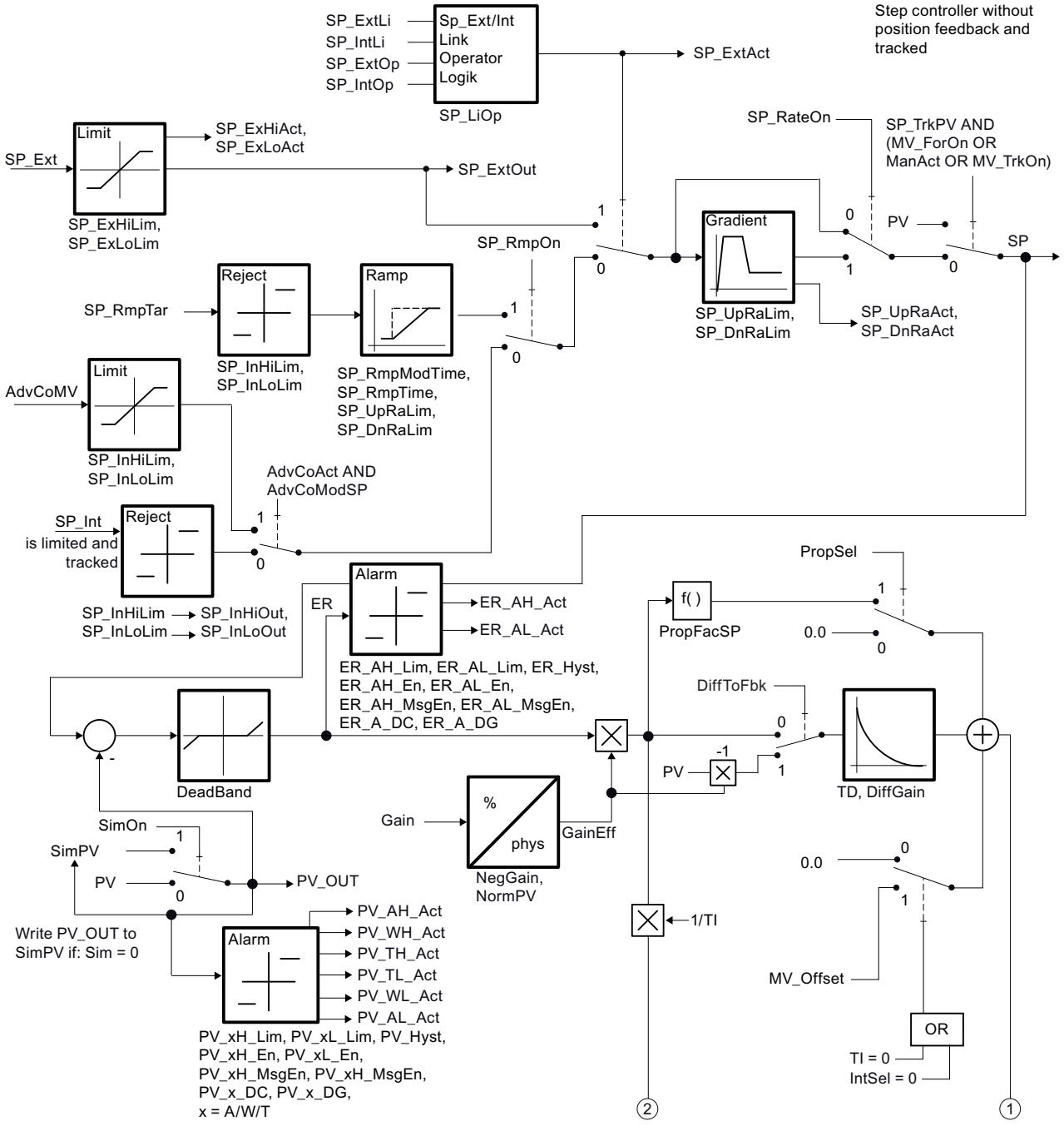
There are two block diagrams for this block: the step controller with and without position feedback.

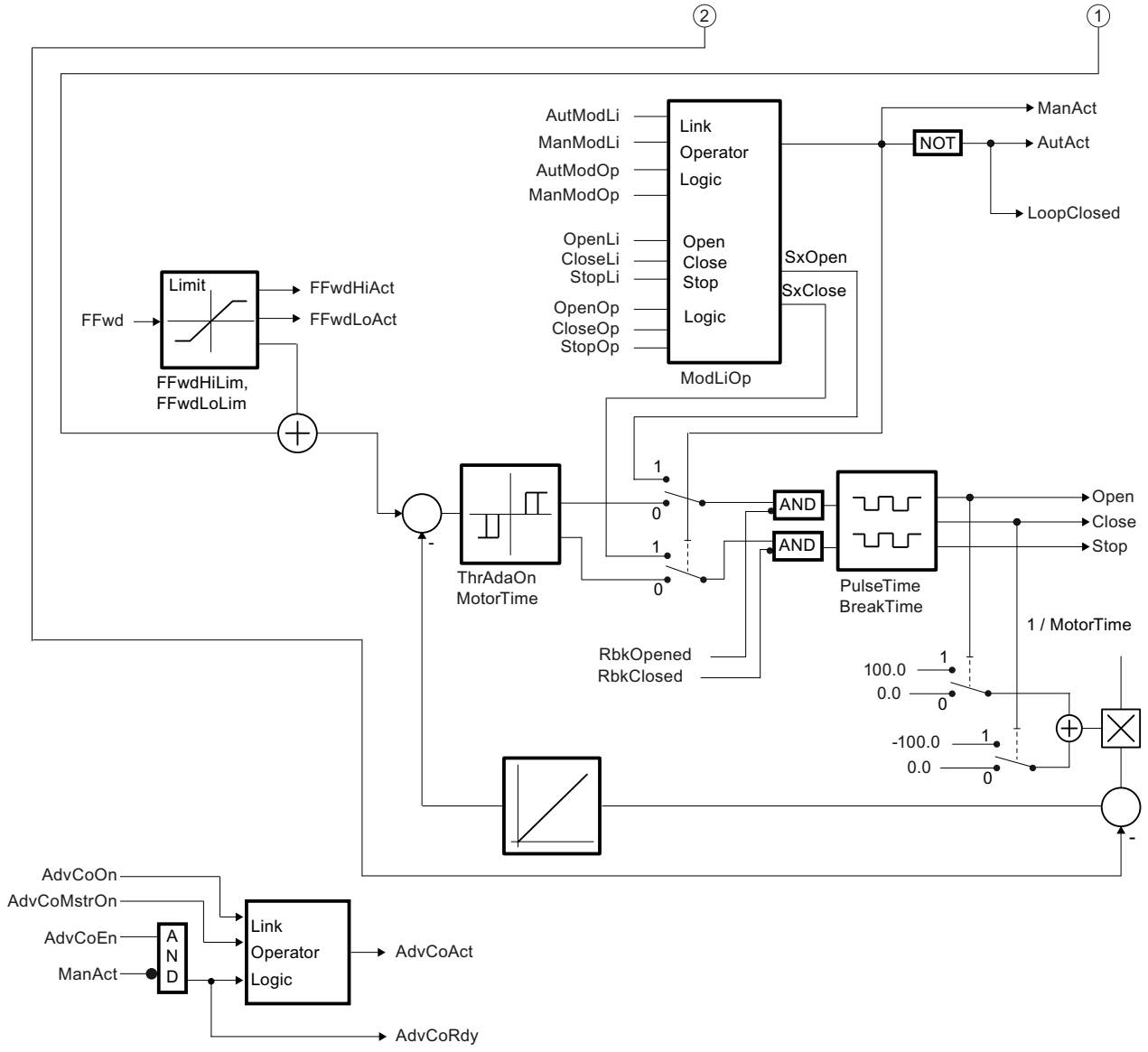
## Step controller with position feedback




1
NOT
AutAct
ManAct
AdvCoAct
AND NOT
AdvCoModSP
AutModLi
ManModLi
AutModOp
ManModOp
Link
Operator
Logic
ModLiOp
O
R
Loop Closed
MV_TrkOn
MV_ForOn
AdvCoMV
MV_Trk
MV_TrkOn
1
1
ER
0
0
Limit
MV_HiAct,
MV_LoAct
MV_HiLim,
MV_LoLim
ConZone,
MV_HiLim,
MV_LoLim
MV_ForOn
MV_Force
1
0
MV
1
0
Reject
Man
is limited
and tracked
ManHiLim
ManLoLim
ManHiOut,
ManLoOut
Limit
FFwd
FFwdHiAct
FFwdLoAct
FFwdHiLim,
FFwdLoLim
Alarm
Rbk
RbkWH_Act
RbkWL_Act
RbkWH_Lim,
RbkWL_Lim,
RbkHyst,
RbkWH_En,
RbkWL_En,
RbkWH_MsgEn,
RbkWL_MsgEn
-
Rbk
ThrAdaOn
MotorTime
AND
AND
PulseTime
BreakTime
Open
Close
Stop
RbkOpened
RbkClosed
AdvCoEn
MV_TrkOn
MV_ForOn
AND
AdvCoModSP
AdvCoModSP
ManAct
OR
AND
AdvCoOn
AdvCoMstrOn
Link
Operator
Logic
AdvCoAct
AdvCoRdy

## Step controller without position feedback






## See also

Description of PIDStepL (Page 1141)
PIDStepL modes (Page 1146)
PIDStepL functions (Page 1147)
PIDStepL error handling (Page 1159)
PIDStepL messaging (Page 1160)
PIDStepL I/Os (Page 1163)

## 6.10.8 Operator control and monitoring

### 6.10.8.1 PIDStepL views

#### Views of the PIDStepL block

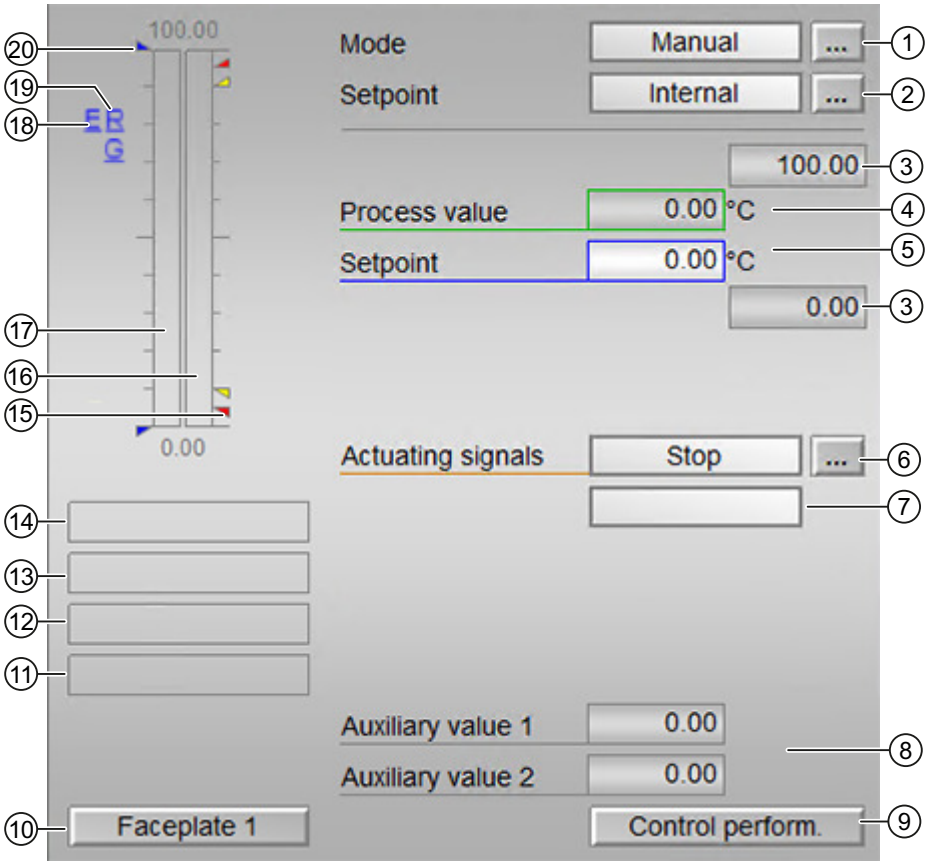
The block PIDStepL provides the following views:

- PIDStepL standard view without position feedback (Page 1185)
- PIDStepL standard view with position feedback (Page 1189)
- Alarm view (Page 336)
- Limit view of PID controllers (Page 325)
- Trend view (Page 341)
- Ramp view (Page 334)
- Parameter view of PID controllers (Page 315)
- PIDStepL preview (Page 1193)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icons for PID and FM controller (Page 269)

Refer to the Structure of the faceplate (Page 279) and Block icon structure (Page 260) sections for general information about the faceplate and block icon.

### 6.10.8.2 PIDStepL standard view without position feedback

#### PIDStepL standard view without position feedback




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 79)
- Automatic mode (Page 79)
- Program mode for controllers (Page 85)
- Out of service (Page 77)

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the operating mode.

### (2) Displaying and switching the setpoint

This area shows how to specify the setpoint. The setpoint can specified as follows:

- By the application ("External", CFC/SFC)
- By the user directly in the faceplate ("Internal").

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the setpoint specification.

You can find additional information on this in the Setpoint specification - internal/external (Page 136) section.

### (3) High and low scale range for the process value

These values provide information on the display range for the bar graph of the process value. The scale range is defined in the Engineering System (ES). For more information refer Bar graph (Page 376) section.

### (4) Display of the process value including signal status

This area shows the current process value with the corresponding signal status.

### (5) Displaying and changing the setpoint including signal status

This area shows the current setpoint with the corresponding signal status.

Refer to the Changing values (Page 291) section for information on changing the setpoint. The setpoint specification also needs to be set to "Internal" for this block.

### (6) Displaying and changing manipulated variables

This area shows you the currently valid manipulated variable. Refer to the Switching operating states and operating modes (Page 289) section for information on changing the manipulated variable.

The following manipulated variables can be selected:

- "Open"
- "Stop"
- "Close"

If text is configured for these commands, it is displayed as status text and as button labels for command selection. You can find more information about this in the Section Labeling of buttons and text (Page 235).

### (7) Displaying the feedback

The following feedback can be displayed:

- "Open"
- "Closed"

If text is configured for these commands, it is displayed as status text and as button labels for command selection. You can find more information about this in the Section Labeling of buttons and text (Page 235).

### (8) Display for auxiliary values

You can use this area to display two auxiliary values that have been configured in the engineering system (ES). You can find more information on this in the section Displaying auxiliary values (Page 239).

### (9) Navigation button for switching to the standard view of the ConPerMon block

Use this navigation button to reach the standard view of the ConPerMon block. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See also the Opening additional faceplates (Page 234) section for more on this.

### (10) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the Engineering System (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Opening additional faceplates (Page 234) section.

### (11) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on the display area for states of the block is available in section Release for maintenance (Page 70).

### (12) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"

You can find additional information on this in the Simulating signals (Page 64) section.

### (13) Display area for block states

This area provides additional information on the operating state of the block:

- "Internal error"
- "Invalid Signal"

### (14) Display area for block states

This area provides additional information on the operating state of the block (from high to low according to priority):

- "Optimization"
- "Tracking"
- "Forced tracking"
- "SP ramp active"

### (15) Limit display

These colored triangles show you the configured limits in the respective bar graph.

### (16) Bar graph for the process value

This area shows the current process value in the form of a bar graph. You can also see the high and low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (17) Bar graph for the setpoint

This area shows the current setpoint in the form of a bar graph. You can also see the high and low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (18) Display of external setpoint

This display [E] is only visible when you have selected "Internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

### (19) Display for the target setpoint of the setpoint ramp

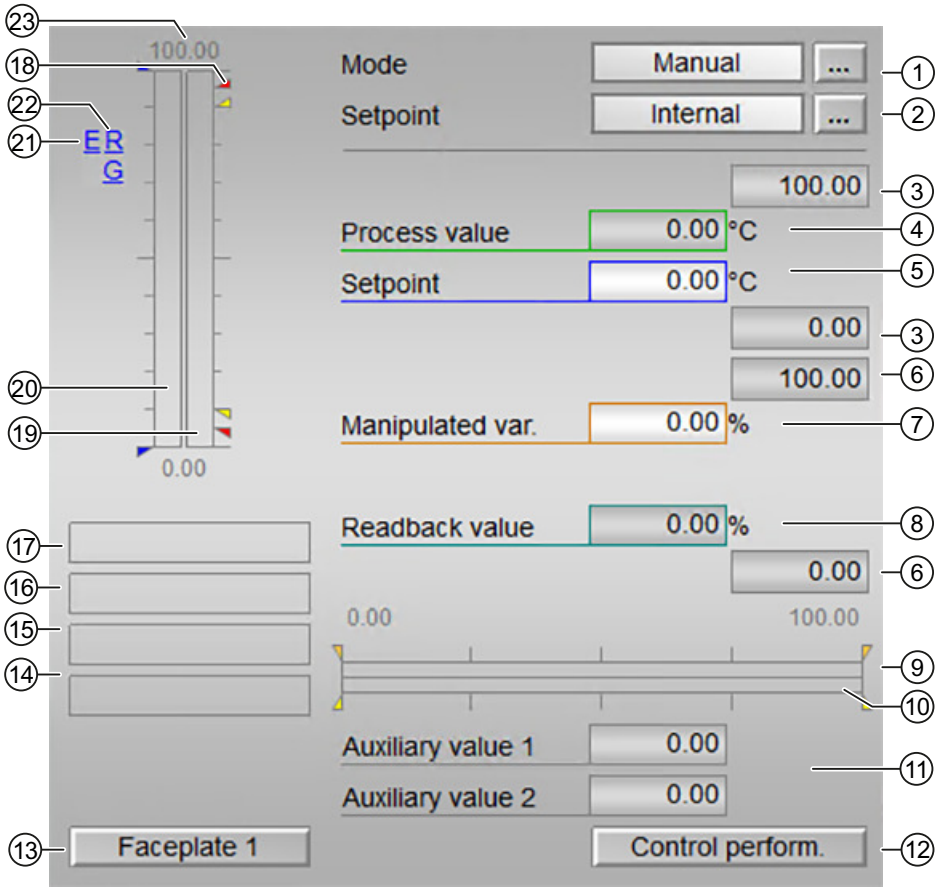
This display [R] shows you the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 334).

### (20) Displaying the limits

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the Engineering System (ES).

### 6.10.8.3 PIDStepL standard view with position feedback

**PIDStepL standard view with position feedback**




**(1) Display and switch the operating mode**

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 79)
- Automatic mode (Page 79)
- Program mode for controllers (Page 85)
- Out of service (Page 77)

Refer to chapter Switching operating states and operating modes (Page 289) for information on switching the operating mode.

### (2) Display and switch the setpoint specification

This area shows how to specify the setpoint. The setpoint can be specified as follows:

- By the application ("External", CFC/SFC)
- By the user directly in the faceplate ("Internal").

Refer to the Switching operating states and operating modes (Page 289) chapter for information on switching the setpoint specification.

You can find additional information on this in section Setpoint specification - internal/external (Page 136).

### (3) High and low scale range for the process value

These values provide information on the display range for the bar graph of the process value. The scale range is defined in the Engineering System (ES). For more information referBar graph (Page 376) section.

### (4) Display of the process value including signal status

This area shows the current process value with the corresponding signal status.

### (5) Display and change the setpoint including signal status

This area shows the current setpoint with the corresponding signal status.

Refer to the Changing values (Page 291) chapter for information on changing the setpoint. The setpoint specification also needs to be set to "Internal" for this block.

### (6) High and low limits of the manipulated variable

You can only display and change a manipulated variable in "manual mode".

### (7) Displaying and changing the manipulated variable

You can only display and change a manipulated variable in "manual mode".

### (8) Displaying the feedback

This display is only visible when the corresponding block input is interconnected.

The following feedback can be displayed:

- "Open"
- "Closed"

### (9) Bar graph for the manipulated variable

The bar graph for the manipulated variable is only available in manual mode. You can also see the high and low limits of the bargraph's scale.

### (10) Bar graph for the readback value

This display is only visible when the corresponding block input is interconnected. You can also see the high and low limits of the bargraph's scale.

The bar graph for the readback value is only available in manual mode.

### (11) Display for auxiliary values

You can use this area to display two auxiliary values that have been configured in the engineering system (ES). You can find more information on this in the section Displaying auxiliary values (Page 239).

### (12) Button for switching to the standard view of the ConPerMon block

Use this button for the standard view of the ConPerMon block. The visibility of this button depends on the configuration in the engineering system (ES).

Refer also to chapter Opening additional faceplates (Page 234) for more on this.

### (13) Button for switching to the standard view of any faceplate

Use this button for the standard view of a block configured in the Engineering System (ES). The visibility of this button depends on the configuration in the engineering system (ES).

You can find additional information on this in section Opening additional faceplates (Page 234).

### (14) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this is available in chapter Release for maintenance (Page 70) Display area for block states.

### (15) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"

You can find additional information on this in section Simulating signals (Page 64).

### (16) Display area for block states

This area provides additional information on the operating state of the block:

- "Internal errorl"
- "Invalid Signal"

### (17) Display area for block states

This area provides additional information on the operating state of the block (from high to low according to priority):

- "Optimization"
- "Tracking"
- "Forced tracking"
- "SP ramp active"

### (18) Limit display

These colored triangles show you the configured limits in the respective bar graph.

### (19) Bar graph for the process value

This area shows the current process value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (20)Display of external setpoint

This display [E] is only visible when you have selected "Internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

### (21) Display for the target setpoint of the setpoint ramp

This display [R] shows the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 334).

### (22) Bar graph for the setpoint

This area shows the current setpoint in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (23) Displaying the limits

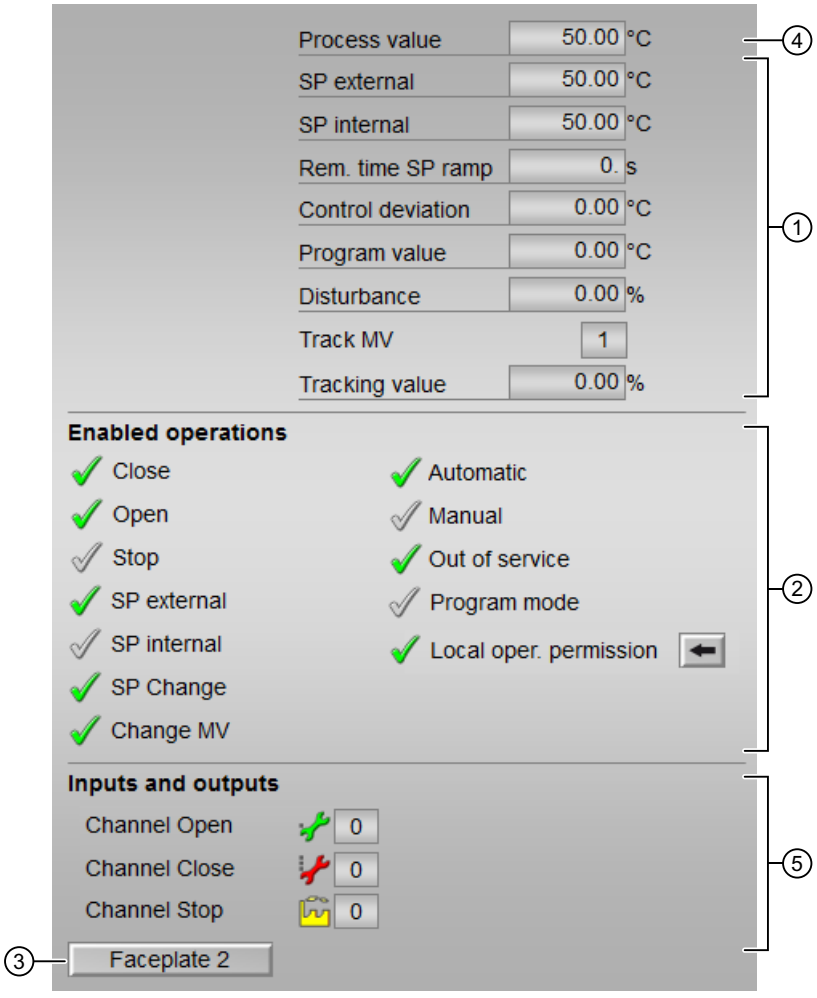
These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the Engineering System (ES).

### See also

Labeling of buttons and text (Page 235)

### 6.10.8.4 PIDStepL preview

#### Preview of PIDStepL




#### (1) Preview area

This area shows you a preview for the following values:

- "SP external": currently applicable external setpoint
- "SP internal": currently applicable internal setpoint
- "Rem. time SP ramp": Remaining time to reach the ramp target value.
- "Error signal": Current control deviation
- "Program value": specified value for program mode
- "Disturbance variable": additive value for feedforward control

- "Track MV": Track manipulated variable (value is 1)
- "Tracking value": effective manipulated variable for "Track manipulated variable"

### (2) Enable operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "Close": You can select the manipulated variable "Close". If text is configured for this command, it is also displayed in brackets. You can find additional information on this in the section Labeling of buttons and text (Page 235) .
- "Open": You can select the manipulated variable "Open". If text is configured for this command, it is also displayed in brackets. You can find additional information on this in the section Labeling of buttons and text (Page 235) .
- "Stop": You can select the manipulated variable "Stop". If text is configured for this command, it is also displayed in brackets. You can find additional information on this in the section Labeling of buttons and text (Page 235) .
- "SP external": You can feedforward the external setpoint.
- "SP internal": You can feedforward the internal setpoint.
- "Change SP": You can change the setpoint.
- "Change MV": You can change the manipulated variable.
- "Program mode": You can switch to "program mode".
- "Automatic": You can switch to "automatic mode".
- "Manual": You can switch to "manual mode".
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operator permission": Use the ← button to switch to the standard view of the OpStations block. You can find additional information on this in the section Operator control permissions (Page 285) .

### (3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to open the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234) .

### (4) Process value

This area displays the real process value (PV).

### (5) Display of the current control signal

This area shows the most important parameters for this block with the current selection

- "Channel Open": Signal from the output channel block for "Open"
- "Channel Close": Signal from the output channel block for "Close"
- "Channel Stop": Signal from the output channel block for "Stop"

# 6.11 Ratio - Ratio controlling

## 6.11.1 Description of Ratio

### Object name (type + number) and family

Type + number: FB 1883

Family: Control

### Area of application for Ratio

The block is used for the following applications:

- Forming a ratio

### How it works

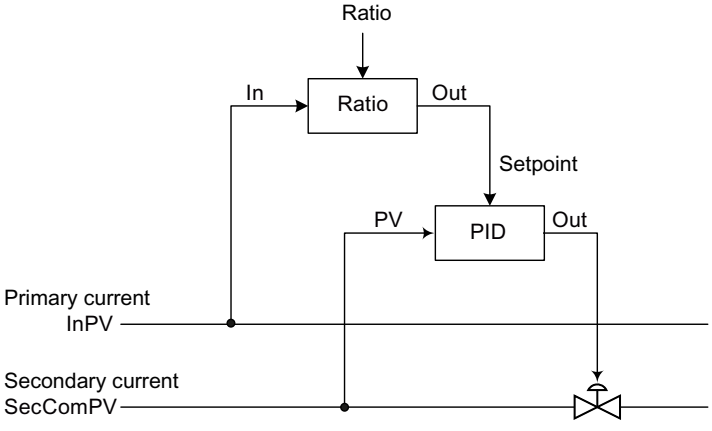
The block is used to create a ratio, for example, in a ratio control. It is also used to set elements (for example, synchronization control loop), or to influence the reference variable of a cascade.

The block operates according to the equation: `Out` = `In` · `RatioOut` + `Offset`

Where:

- `RatioOut` is either the value from `RatioInt` or `RatioExt`
- `Offset` is the offset value that should be added to the output value

`In` is based on the interconnection whereas `RatioOut` is selected based on the internal/ external selection.




Also refer to the Ratio control with control loop monitoring through ConPerMon (RatioControl) (Page 2836) section for information on calculating the current ratio.

### Configuring OBs

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

### Status word allocation for `Status` parameter

The description for each parameter can be found in the Ratio I/Os (Page 1202) section.

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | Not used |
| 6 | `OnAct.Value` |
| 7 | Not used |
| 8 | `RatExtAct.Value` |
| 9 | `RatLoAct.Value` |
| 10 | `RatHiAct.Value` |
| 11 | `OutLoAct.Value` |
| 12 | `OutHiAct.Value` |
| 13 | `RatTrkExt` |
| 14 | `SimLiOp.Value` |
| 15 - 29 | Not used |
| 30 | Invalid signal |
| 31 | Internal error |

### See also

Ratio functions (Page 1198)

Ratio messaging (Page 1202)

Ratio block diagram (Page 1207)

Ratio error handling (Page 1201)

Ratio modes (Page 1198)

## 6.11.2 Ratio modes

### Ratio operating modes

The block can be operated using the following modes:

- On (Page 77)
- Out of service (Page 77)

### "On"

General information on the "On" mode is available in the section On (Page 77).

### "Out of service"

You can find general information about the "Out of service" mode in the Out of service (Page 77) section.

The last valid value is provided at the `Out` output in this operating mode. The `OutHiAct`, `OutLoAct`, `RatHiAct`and `RatLoAct`output parameters are reset.

### See also

Ratio block diagram (Page 1207)

Ratio I/Os (Page 1202)

Ratio messaging (Page 1202)

Ratio error handling (Page 1201)

Ratio functions (Page 1198)

Description of Ratio (Page 1196)

## 6.11.3 Ratio functions

### Functions of Ratio

The functions for this block are listed below.

### Internal or external ratio

You can use the RatLiOp parameter to specify if the ratio should be specified internally or externally:

- RatLiOp = 0: The ratio (RatioInt) is specified by the operator. The operator can decide if the specification should be made internally (RatIntOp = 1) or externally (RatExtOp = 1) .
- RatLiOp = 1: The ratio (RatioExt) is specified by an interconnection. The interconnection is used to determine if the specification should be made internally (RatIntLi = 1) or externally (RatExtLi = 1) .

### Bumpless switchover from external to internal ratio

The parameter RatTrkExt = 1 is used so that the internal ratio tracks the external setpoint to achieve a bumpless switchover from the external to the internal ratio. This allows unwanted jumps at the output parameter to be avoided.

### Limiting the ratio

The ratio is limited by the RatHiLim parameter (high) or RatLoLim parameter (low).

The external ratio RatioExt is limited to the value you have specified if a limit is violated. This is then also used to form the output value Out. If these limits are reached or violated RatioHiAct or RatioLoAct are also set to 1.

The internal ratio RatioInt is checked against the value you have specified. If a value is outside the specified limit, it is reset to the most recent valid value.

### Limiting the output value

The output value is limited by the OutHiLim parameter (high) or OutLoLim parameter (low).

The OutHiAct or OutLoAct output parameter is set to 1 as soon as the output value has reached or exceeded the limits

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Analog input value (SimIn, SimInLi)

### Display and operator input area for process values and setpoints

This block provides the standard function Display and operator input area for process values and setpoints (Page 233).

### Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 240).

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can switch to external |
| 5 | 1 = Operator can switch to internal |
| 6 | 1 = Operator can specify the internal ratio |
| 7 - 9 | Not used |
| 10 | 1 =Operator can change the simulation value `SimIn` |
| 11 | 1 = Operator can switch to "Simulation" mode |
| 12 | 1 = Operator can enable bumpless switchover |
| 13 | Not used |
| 14 | 1 = Operator can change `RatHiLim` |
| 15 | 1 = Operator can change `RatLoLim` |
| 16 | 1 = Operator can change `OutHiLim` |
| 17 | 1 = Operator can change `OutLoLim` |
| 18 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Forming and outputting signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` is formed from the following parameters:

- `Out.ST`
- `SecComPV.ST`
- `InPV.ST`
- `RatioExt.ST`
- `Offset.ST`

The signal status of the `Out` output parameter corresponds to the input parameter `In`.

The signal status of the current ratio calculation corresponds to the status of the input parameter `SecComPV`.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 24 | Enabling local operator authorization (Page 169) |

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of Ratio (Page 1196)

Ratio messaging (Page 1202)

Ratio I/Os (Page 1202)

Ratio block diagram (Page 1207)

Ratio error handling (Page 1201)

Ratio modes (Page 1198)

## 6.11.4 Ratio error handling

### Ratio error handling

Refer to section Error handling (Page 128) for basic instructions on how to troubleshoot all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when implementing the block; block will not be processed. |
| 0 | No active fault |
| 30 | The input value of `In` can no longer be displayed in the REAL number field. |
| 51 | `RatLiOp` = 1 and `RatExtLi` = 1 and `RatInLi` = 1 |

### See also

Ratio block diagram (Page 1207)
Ratio I/Os (Page 1202)
Ratio messaging (Page 1202)
Ratio functions (Page 1198)
Ratio modes (Page 1198)
Description of Ratio (Page 1196)

## 6.11.5 Ratio messaging

### Messaging

This block does not offer messaging.

### See also

Description of Ratio (Page 1196)
Ratio functions (Page 1198)
Ratio I/Os (Page 1202)
Ratio block diagram (Page 1207)
Ratio error handling (Page 1201)
Ratio modes (Page 1198)

## 6.11.6 Ratio I/Os

### I/Os of Ratio

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

## Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of block-type<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 1198) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| In | Analog input | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| InPV | Input for process variable | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| InUnit | Unit of measure for input parameter In | INT | 1001 |
| Offset | Offset | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operating permission, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OS_Perm | I/O for operator control permissions (Page 1198) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OutHiLim | High limit for output value | REAL | 100.0 |
| OutLoLim | Low limit for output value | REAL | 0.0 |
| RatExtLi | 1 = Select external ratio (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RatExtOp* | 1 = Select external ratio (via operator) | BOOL | 0 |
| RatHiLim | High limit | REAL | 100.0 |
| RatIntLi | 1 = Select internal ratio (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RatIntOp* | 1 = Select internal ratio (via operator) | BOOL | 1 |
| RatioExt | External ratio | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |
| RatioInt* | Internal ratio | REAL | 1.0 |
| RatioUnit | Unit of measure for the RationInt, RatioExt input parameter or RatioPV (output parameter) | INT | 0 |
| RatLiOp | Select ratio source (internal/external):<br>1 = Via interconnection<br>0 = Via operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RatLoLim | Low limit | REAL | 0.0 |
| RatOpScale | Limit for scale in bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| RatTrkExt | 1 = Bumpless switchover from external to internal ratio active | BOOL | 0 |
| SecComPV* | Process value of secondary component | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SecComUnit | Unit of measure for input parameter SecComPV | INT | 1001 |
| SelFp1 | 1 = Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SelFp2 | 1 = Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | - |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SimIn* | Analog input value used for SimOn = 1 | REAL | 0.0 |
| SimInLi | Value that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see Ratio error handling (Page 1201) | INT | -1 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| Out | Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| OutHiAct | 1 = High limit overshoot | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OutHiLmOut | Output of high limit | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| OutLoAct | 1 = Low limit overshoot | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OutLoLmOut | Output of low limit | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RatExtAct | 1 = External ratio used<br>0 = Internal ratio used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RatHiAct | 1 = Limit (high) for ratio active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RatioPV | Current ratio | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RatioOut | Applied ratio (RatioInt or RatioExt) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RatOpScaleOut | RatOpScale as output | STRUCT<br>High: REAL<br>Low: REAL | |
| RatLoAct | 1 = Limit (low) for ratio active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RatioUnitOut | RatioUnit as output | INT | 0 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status | Status word (Page 1196) | DWORD | 16#00000000 |

## See also

Ratio messaging (Page 1202)
Ratio block diagram (Page 1207)
Ratio modes (Page 1198)

## 6.11.7 Ratio block diagram

### Ratio block diagram

A block diagram is not provided for this block.

### See also

Ratio I/Os (Page 1202)

Ratio messaging (Page 1202)

Ratio error handling (Page 1201)

Ratio functions (Page 1198)

Ratio modes (Page 1198)

Description of Ratio (Page 1196)

## 6.11.8 Operator control and monitoring

### 6.11.8.1 Ratio views

### Views of the Ratio block

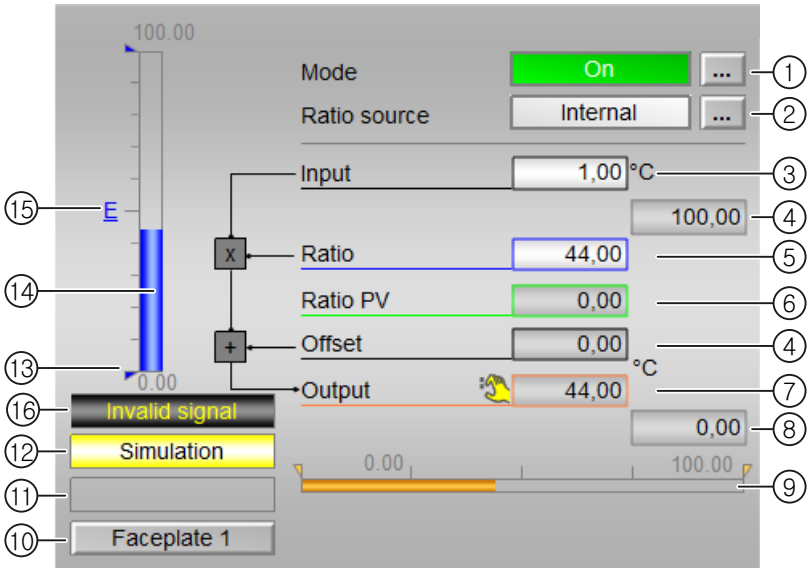
The block Ratio provides the following views:

- Ratio standard view (Page 1208)
- Trend view (Page 341)
- Ratio parameter view (Page 1211)
- Ratio preview (Page 1212)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for Ratio (Page 1213)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 6.11.8.2 Ratio standard view

#### Ratio standard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 77)
- Out of service (Page 77)

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the operating mode.

#### (2) Displaying and switching the ratio source

This area shows you the currently valid signal source for the ratio setpoint. The following signal sources can be shown here:

- "External"
- "Internal"

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the signal source.

#### (3) Displaying the input value

The faceplate is a schematic representation of the function of the Ratio block as a signal flow chart:

Output = Input · Ratio + `Offset`

The input is typically the flow setpoint or actual value of the primary component of a ratio controller.

### (4) High and low scale range for the ratio

The scale range is based on the bar graph for the ratio specification. For more information refer Bar graph (Page 376) section.

### (5) Displaying and switching the default ratio

This area shows you the currently valid specification for the ratio.

The ratio source **(2)** needs to be set to "internal" in order to change this value.

Refer to the Changing values (Page 291) section for information on changing this value.

### (6) Displaying the ratio `PV`

This area shows you the current ratio actual value with the corresponding signal status, i.e. the ratio of the actually measured `PV` from the active controller. The task of the ratio controller is to set the flow of all components so that the actual ratio approximates the specified ratio as closely as possible.

### (7) Display of the `Offset`

This area shows the current `Offset`.

### (8) Displaying the output value

This area shows the current output value `Out`, which typically serves as the setpoint for the flow of the secondary component.

### (9) Bar graph for the output

This display graphically represents the output value with the limits set in the Engineering System (ES) (orange triangles, output parameters `OutHiLmOut` and `OutLoLmOut`). You can also see the high and low limits of the bargraph's scale.

### (10) Displaying the limit

This status display is based on the limit of the output value `Out`.

### (11) Display area for block states

This area shows if the output value has violated the range limits:

- "Output ≥ HL"
- "Output ≤ LL"

You can set the range limits in the parameter view (Page 1211) of the block.

### (12) Display area for block states

This area provides additional information on the operating state of the block:

- Simulation

You can find additional information on this in the Simulating signals (Page 64) section.

### (13) Display of the limits for the ratio

This blue triangle shows the configured range limits for the ratio.

### (14) Bar graph for the default ratio

This area shows you the currently valid specification for the ratio in the form of a bar graph. The visible area in the bar graph depends on the configuration of the ratio in the engineering system (ES). You can also see the high and low limits of the bargraph's scale.

### (15) Display of external setpoint

This display [E] is only visible when you have selected "Internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

### (16) Display area for block states

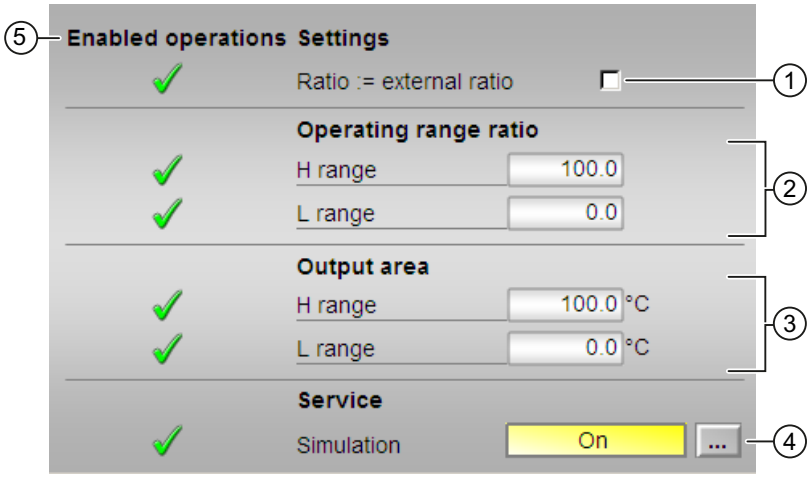
This area provides additional information on the operating state of the block:

- Internal error
- Invalid Signal

You can find additional information on this in Error handling (Page 128) section.

### 6.11.8.3 Ratio parameter view

#### Parameter view of Ratio




#### (1) Settings Ratio := external ratio

When the check box is selected ☑, the ratio is bumplessly switched from external to internal. The internal ratio value is tracked to the external one.

#### (2) Operating range ratio

This is where you specify the operating range for the ratio (input parameters `RatHiLim` or `RatLoLim`). The range is indicated by a blue triangle in the standard view of the bar graph.

#### (3) Range for output value

This is where you specify the operating range for the output value (input parameters `OutHiLmOut` or `OutLoLmOut`).

#### (4) Service

You can select the following functions in this area:

- "Simulation"

You can find information about this in the following chapters:

- Switching operating states and operating modes (Page 289)
- Simulating signals (Page 64)
- Release for maintenance (Page 70).

### (5) Enabled operations

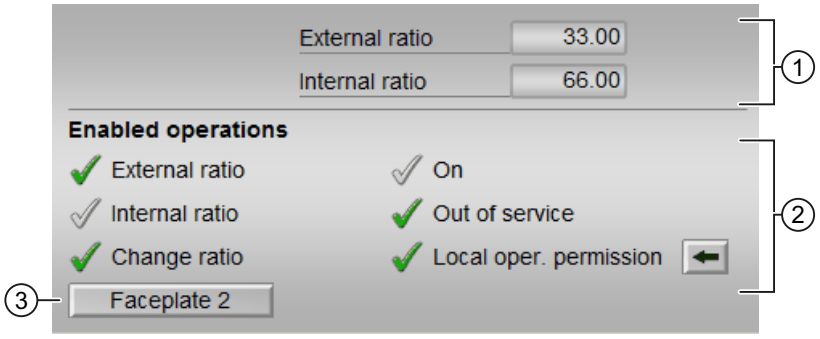
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

## 6.11.8.4 Ratio preview

### Preview of Ratio




### (1) Preview area

This area shows you a preview for the following values:

- "External ratio": currently applicable external ratio
- "Internal ratio": currently applicable internal ratio

### (2) Enabled operations

This area shows all operations for which special operating permissions are assigned. They depend on the configuration in the Engineering System (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter at all due to the configured AS operating permissions (`OS_Perm` or `OS1Perm`).

The following enabled operations are shown here:

- "Ratio external": You can change the external ratio.
- "Ratio internal": You can change the internal ratio.
- "Change ratio": You can change the ratio.
- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. You can find additional information on this in chapter Operator control permissions (Page 285).

**(3) Navigation button for switching to the standard view of any faceplate**

Use this navigation button to reach the standard view of a block configured in the Engineering System (ES). The visibility of this navigation button depends on the configuration in the Engineering System (ES).

You can find additional information on this in chapter Opening additional faceplates (Page 234).

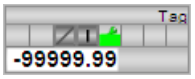
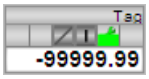
### 6.11.8.5 Block icon for Ratio

**Block icons for Ratio**

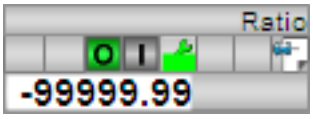
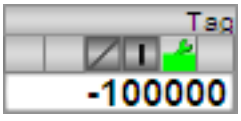
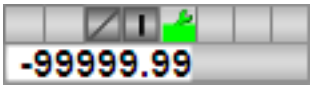
A variety of block icons are available with the following functions:

- Process tag type
- Display of the ratio
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits
- Operating modes
- Internal and external setpoint specification
- Signal status, release for maintenance
- Displays for bypassing interlocks
- Interlocks
- Memo display

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | |
|  | 6 | |
|  | 7 | |
|  | 8 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269).

# 6.12 SplRange - Signal splitter

## 6.12.1 Description of SplRange

### Object name (type + number) and family

Type + number: FC 372

Family: Control

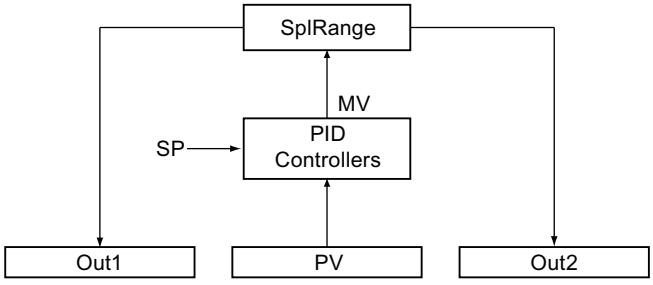
### Area of application for SplRange

The block is used for the following applications:

- Splitting the output signal of a PID controller

### How it works

The block is used to split signals output coming from a PID controller. You can use this controller to control two valves, for example.




The output parameters are calculated as follows:

`Out1` = `Out1Scale.Low` + ((`NeutPos` - `DeadBand` - `In`) / (`NeutPos` - `DeadBand` - `InScale.Low`)) · (`Out1Scale.High` - `Out1Scale.Low`)

`Out2` = `Out2Scale.Low` + ((`NeutPos` + `DeadBand` - `In`) / (`NeutPos` + `DeadBand` - `InScale.High`)) · (`Out2Scale.High` - `Out2Scale.Low`)

The neutral position (`NeutPos`) forms the reference point for selecting the individual splitter profiles. For details, refer to the SplRange functions (Page 1218) section.

Refer to SplRange I/Os (Page 1222) to learn the meaning of the parameters.

The block is installed in the run sequence downstream of the controller block. The manipulated variable output (`MV`) of the controller block is interconnected to the input `In` of the SplRange block.

The neutral position (`NeutPos`) and the dead band zone (`DeadBand`) can be set by means of the corresponding parameters. `DeadBand` must be configured to be less than `NeutPos`.

`Out1` and `Out2` are adapted to the physical variable by configuring the high/low limits of `Out1` and `Out2`.

The Out1Act or Out2Act output parameter indicates (= 1) that the corresponding Out1 or Out2 output parameter is enabled if the In input value is less than (for Out1) the neutral position (NeutPos) or the In input value is greater than the neutral position (NeutPos) depending on the dead band (Deadband).

### Configuration

Use CFC editor to install the block in the OB in which the controller block runs whose manipulated variable is being processed.

There are templates (Templates) for the SplRange block for process tag types in the Advanced Process Library, one example (APL_Example_xx, xx refers to the language variant) with an application scenario for this block being:

- Split-range controller with control loop monitoring through ConPerMon (SplitrangeControl) (Page 2833)

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for Status parameter

This block does not have the Status parameter.

### See also

SplRange messaging (Page 1221)

SplRange block diagram (Page 1223)

SplRange error handling (Page 1221)

SplRange modes (Page 1217)

## 6.12.2 SplRange modes

### SplRange operating modes

This block does not have any modes.

### See also

SplRange I/Os (Page 1222)

SplRange messaging (Page 1221)

SplRange error handling (Page 1221)

SplRange functions (Page 1218)

Description of SplRange (Page 1216)
SplRange block diagram (Page 1223)

## 6.12.3 SplRange functions

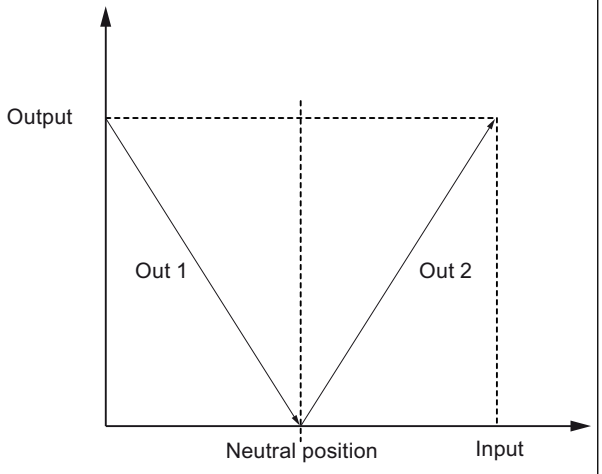
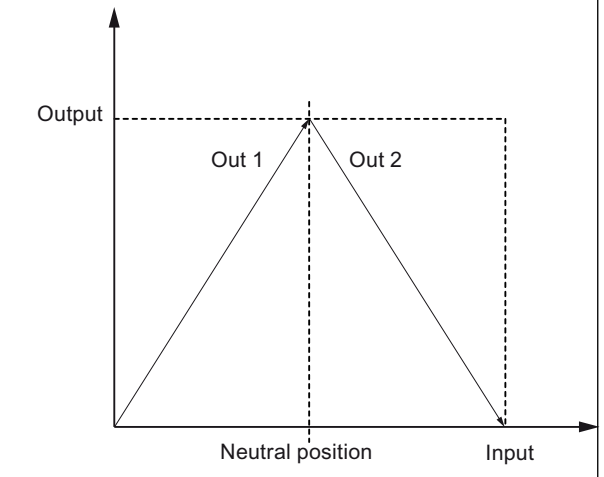
### Functions of SplRange

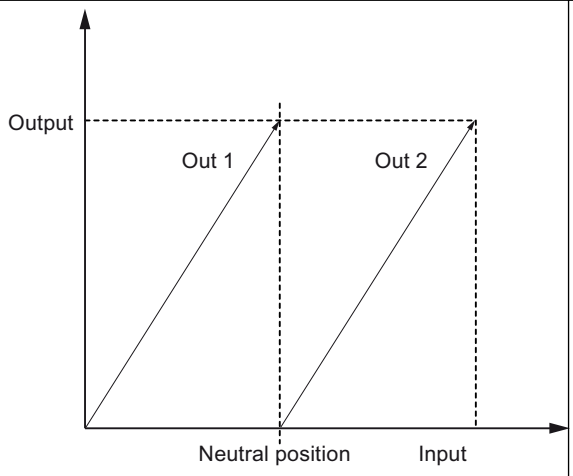
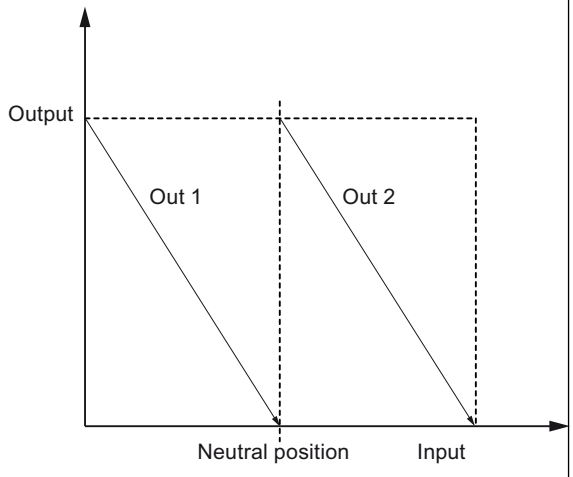
The functions for this block are listed below.

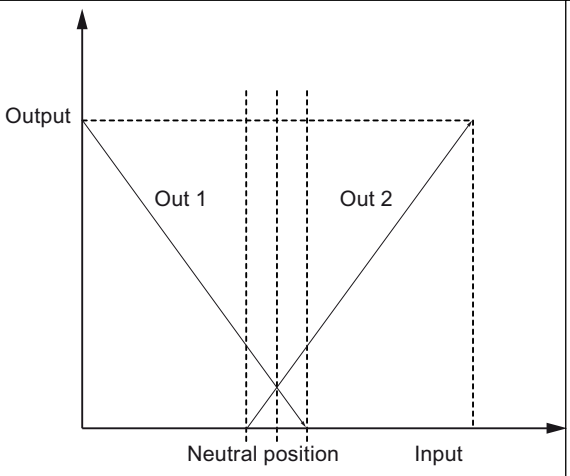
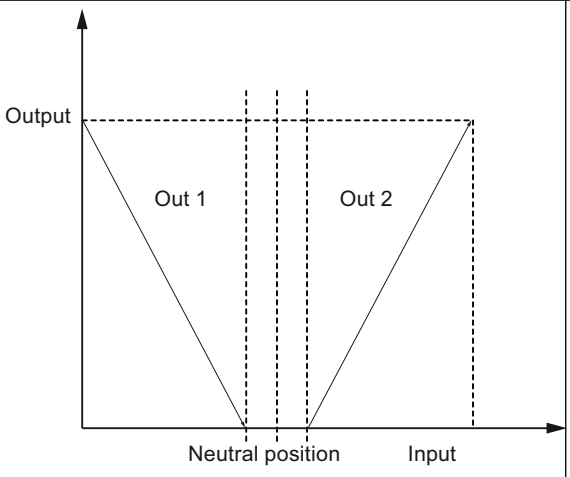
### Splitting of the output signal of a controller

The `In` block input can be used to split the output signal of a controller.

The table below shows the six types of signal splitting that are available:

| Case | Signal splitting | Setting the I/Os |
|---|---|---|
| 1 |  | `Out1Scale.Low` = 0.0 and `Out1Scale.High` = 100.0<br>`Out2Scale.Low` = 0.0 und `Out2Scale.High` = 100.0<br>`DeadBand` = 0.0 and `NeutPos` = 50.0 |
| 2 |  | `Out1Scale.Low` = 100.0 and `Out1Scale.High` = 0.0<br>`Out2Scale.Low` = 100.0 and `Out2Scale.High` = 0.0<br>`DeadBand` = 0.0 and `NeutPos` = 50.0 |

| Case | Signal splitting | Setting the I/Os |
|---|---|---|
| 3 |  | `Out1Scale.Low` = 100.0 and `Out1Scale.High` = 0.0<br>`Out2Scale.Low` = 0.0 and `Out2Scale.High` = 100.0<br>`DeadBand` = 0.0 and `NeutPos` = 50.0 |
| 4 |  | `Out1Scale.Low` = 0.0 and `Out1Scale.High` = 100.0<br>`Out2Scale.Low` = 100.0 and `Out2Scale.High` = 0.0<br>`DeadBand` = 0.0 and `NeutPos` = 50.0 |

| Case | Signal splitting | Setting the I/Os |
|---|---|---|
| 5 |  | `Out1Scale.Low` = 0.0 and `Out1Scale.High` = 100.0<br>`Out2Scale.Low` = 0.0 and `Out2Scale.High` = 100.0<br>`DeadBand` = -10.0 and `NeutPos` = 50.0 |
| 6 |  | `Out1Scale.Low` = 0.0 and `Out1Scale.High` = 100.0<br>`Out2Scale.Low` = 0.0 and `Out2Scale.High` = 100.0<br>`DeadBand` = 10.0 and `NeutPos` = 50.0 |

## See also

Description of SplRange (Page 1216)
SplRange messaging (Page 1221)
SplRange I/Os (Page 1222)
SplRange block diagram (Page 1223)
SplRange error handling (Page 1221)
SplRange modes (Page 1217)

## 6.12.4 SplRange error handling

### Error handling of SplRange

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 30 | The value of `In` can no longer be displayed in the REAL number field. |

### See also

SplRange block diagram (Page 1223)

SplRange I/Os (Page 1222)

SplRange messaging (Page 1221)

SplRange functions (Page 1218)

SplRange modes (Page 1217)

Description of SplRange (Page 1216)

## 6.12.5 SplRange messaging

### Messaging

This block does not offer messaging.

### See also

Description of SplRange (Page 1216)

SplRange functions (Page 1218)

SplRange I/Os (Page 1222)

SplRange block diagram (Page 1223)

SplRange error handling (Page 1221)

SplRange modes (Page 1217)

## 6.12.6 SplRange I/Os

### I/Os of SplRange

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| DeadBand | Width of dead band | REAL | 0.0 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| In | Input value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| InScale | Limit range for the input signal | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| NeutPos | Neutral position | REAL | 50.0 |
| Out1Scale | Limit range of output 1 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| Out2Scale | Limit range of output 2 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see SplRange error handling (Page 1221). | INT | -1 |
| Out1 | Output value 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Out1Act | 1 = Output 1 is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Out2 | Output value 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Out2Act | 1 = Output 2 is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### See also

Description of SplRange (Page 1216)
SplRange functions (Page 1218)
SplRange messaging (Page 1221)
SplRange block diagram (Page 1223)
SplRange modes (Page 1217)

## 6.12.7 SplRange block diagram

### SplRange block diagram

A block diagram is not provided for this block.

### See also

SplRange I/Os (Page 1222)
SplRange messaging (Page 1221)
SplRange error handling (Page 1221)
SplRange functions (Page 1218)
Description of SplRange (Page 1216)
SplRange modes (Page 1217)

# 6.13 AutoExcitation - Process trigger for predictive controller

## 6.13.1 Description of AutoExcitation

### Object name (type + number) and family

Type + number: FB 1842

Family: Control

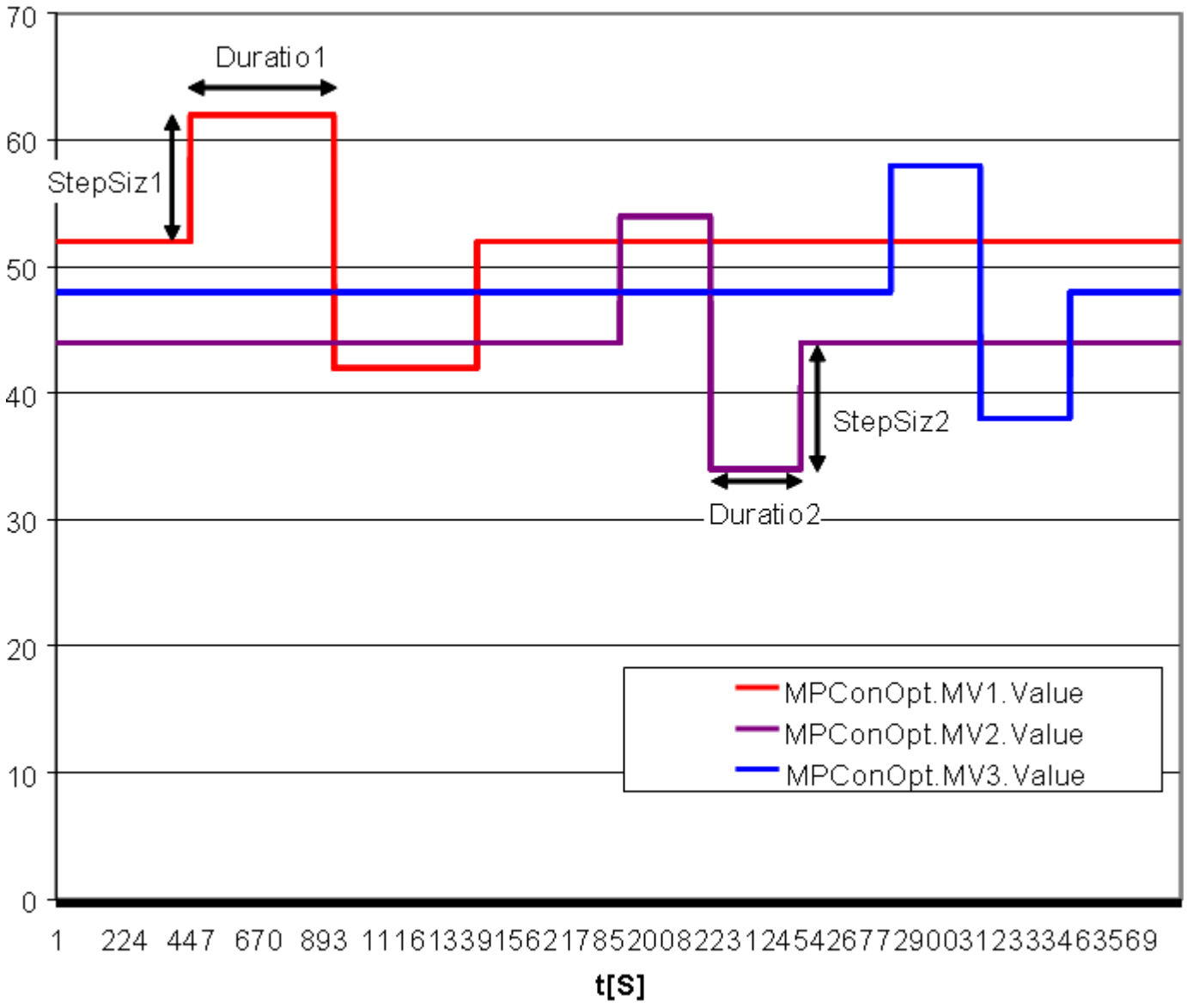
### Area of application of AutoExcitation

This block is used to generate suitable excitation signals for the identification of dynamic multi-variable process models for the model-based predictive controllers ModPreCon and the large predictive controller MPC10x10. It is interconnected with aModPreCon or MPC10x10 block for this purpose.

### How it works

The AutoExcitation block generates a sequence of abrupt trigger signals for the selected manipulated variables, MV1, MV2 to NumberMVs. A jump is initially made up, down, and then back to the starting point for each manipulated variable, so that the signals is symmetrical to the operating point.

The operating point taken from assigned controller via the MV1Actual...MV10Actual inputs. You set the jump height for each manipulated variable with the StepSiz1... StepSiz10 parameter and the jump duration via the Duratio1...Duratio10 parameter.




## Configuration

The AutoExcitation  block is part of the new process tag type, ModPreCon and MPC10x10. Only the new controller version has the necessary additional inputs for automatic triggering.

If the block retro-installed in an existing CFC, make sure that AutoExcitation andModPreCon or MPC10x10 are in the same OB cycle.

- Supply the AutoExcitation.MV1Actual...MV10Actual input variables with the appropriate ModPreCon.MV1...MV4 or MPC10x10.MV1...MV10 values from the controller.
- Specify the input parameter AutoExcitation.NumberMVs according to the number of manipulated variables you want to use in your MPC application. Exception: If you do not want to automatically trigger all manipulated variables of the controller, specify the number of manipulated variables to be directly triggered at the AutoExcitation.NumberMVs input parameter .
- Connect the AutoExcitation.MV1Excite...MV10Excite output variables to the appropriate ModPreCon.MV1Excite... MV4Excite or MPC10x10.MV1Excite...MV10Exciteinput variables on the controller.
- Connect the AutoExcitation.ExciteAct output variable to the ModPreCon.ExciteOn or MPC10x10.ExciteOn input variables of the controller.

## 6.13.2 AutoExcitation modes

### Operating modes of AutoExcitation

This block does not have any modes.

## 6.13.3 AutoExcitation functions

### Using AutoExcitation

Start triggering by setting the StartExcite binary input variable to the value 1 .

If you cancel triggering during ongoing operation via StartExcite =0 , the manipulated variables are reset to the original operating point.

When the Next binary input parameter = 1 , you can start the next jump manually before the planned time has expired.

The StepPhase output variable shows the current phase of the attempted jump:

0: Constant

1: Jump up

2: Jump down

3: Jump back to the operating point

The StepTime output variable indicates the remaining time in [s] until the next jump.

## 6.13.4 AutoExcitation error handling

### Overview of error numbers

The ErrorNum output parameter can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Default value when the block is installed, the block is not executed |
| 0 | There is no error. |
| 60 | One of the input parameters, Duratio1,2,3,4...10 < SampleTime or SampleTime < 0.001 |
| 61 | The trigger signals are not adopted correctly by the assigned controller. |

## 6.13.5 AutoExcitation messaging

### Messaging of AutoExcitation

This block does not offer messaging.

## 6.13.6 AutoExcitation I/Os

### I/Os of AutExcite

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Duration1...Duration10 | Duration for jump in MV1...MV10 in s | REAL | 20 |
| MV1Actual...MV10Actual | Current value of MV1...MV10 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Next* | 1 = Manual start of the next trigger change | BOOL | 0 |
| SampleTime | Sampling time in s | REAL | 1 |
| StartExcite* | 1 = Start of automatic triggering; 0= Cancel automatic triggering | BOOL | 1 |
| StepSize1...StepSize10 | Step height of MV1...MV10 | REAL | 10 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ErrorNum | Output of pending error number | INT | 0 |
| ExciteAct | 1 = Automatic triggering is active | BOOL | 0 |
| MV1Excitation...MV10Excitation | Value of the trigger for MV1...MV10 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| StepPhase | Phase of the jump; 0 = Constant, 1= Jump up, 2 = Jump down, 3 = Return | INT | 0 |
| StepTime | Time remaining until the next jump phase in s | REAL | 0 |

## 6.13.7 AutoExcitation block diagram

### Block diagram of AutoExcitation

A block diagram is not provided for this block.

# 6.14 LPOptim - Optimization after traversing the linear programming

## 6.14.1 Description of LPOptim

### Object name (type + number) and family

Type + number: FB1844

Family: Control or system

### Area of application of LPOptim

Optimization of a linear performance function taking into account constraints after traversing the linear programming.

The LPOptim block is used by the ModPreCon block for static operating point optimization (Page 976). However, it can also be used separately.

### Operating principle

The performance function has a linear relationship to up to four decision variables, x1...x4, with corresponding weighting factors, g1...g4 .

J = g1*x1 + g2*x2 + g3*x3 + g4*x4

The performance function is a hyper-level in a five dimensional space whose gradient direction is described by the gradient vector, Gradient = [ g1, g2, g3, g4].

The block can determine both the maximum as well as the minimum of the J performance function. A minimization problem is transformed into an equivalent maximization problem by inverting the gradient vector.

The decision variables are positive and limited by a system of up to eight linear non-equivalence constraints .

The first constraint is, for example

a11*x1+a12*x2+a13*x3+a14*x4 <= b1

All four elements of the first line from the A matrix are combined in the A1n data structure. All eight elements of the B vector are combined in the Begrenzungen data structure.

### How it works

The block operates according to the simplex procedure, i.e. based on an iterative algorithm. Due to the linear performance criterion, the solution always lies at an intersection of multiple restrictions.

### Configuration

The block can be installed in OB1 because no time-critical calculations are performed. If the block is called by ModPreCon , the user does not have to set parameters directly at LPOptim block.

## 6.14.2 LPOptim modes

### Operating modes of LPOptim

This block does not have any modes.

## 6.14.3 LPOptim functions

### Use of LPOptim

The algorithm is started with a positive edge at the Start parameter or if there is a change in one of the input variables (gradient vector and limitations), and displays the optimum variable found at the XOpt1...XOpt4 output parameters when execution is completed.

If no solution is found (non-plausible coefficient), execution is aborted with an error code.

## 6.14.4 LPOptim error handling

### Overview of error numbers

The Status output parameter can be used to display the following states:

| Status | Meaning |
|---|---|
| 0 | There is no error. |
| 10 | Unrestricted target function |
| 15 | No XOpt found |
| 20 | Infinite number of solutions |
| 35 | Unlimited permissible range |
| 40 | Configuration error |

## 6.14.5 LPOptim messaging

### Messaging

This block does not offer messaging.

## 6.14.6 LPOptim I/Os

I/Os of LPOptim

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

## I/Os of LPOptim

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| J_Mini | 1= Search minimum<br>0= Search maximum | BOOL | 0 |
| Start* | 1= Start optimization calculation immediately<br>0= Start optimization calculation only if one of the input variables has changed | BOOL | 0 |
| X1 | Decision variable 1 for calculation of J_Act | REAL | 0 |
| X2 | Decision variable 2 for calculation of J_Act | REAL | 0 |
| X3 | Decision variable 3 for calculation of J_Act | REAL | 0 |
| X4 | Decision variable 4 for calculation of J_Act | REAL | 0 |
| J_0 | Constant part of the performance criterion, irrespective of the decision variables | REAL | 0 |
| N_X | Number of decision variables,<br>N_X <= 4 | INT | 4 |
| N_b | Number of constraints,<br>N_b <= 8 | INT | 8 |
| Gradient.g1*<br>...<br>Gradient.g4* | Gradient vector with up to 4 REAL elements | STRUCT | [ 1, 1, 1, 1] |
| Constraints.b1<br>...<br>Constraints.b8 | Vector B of the limits with up to 8 REAL elements | STRUCT | [ 0, 0, 0, 0, 0, 0, 0, 0] |
| A1n.a11<br>A1n.a12<br>A1n.a13<br>A1n.a14 | 1. Line of the A matrix contains factors for the elements of vector X in the first constraint | STRUCT | [ 0, 0, 0, 0] |
| A2n.a21<br>A2n.a22<br>A2n.a23<br>A2n.a24 | 2. Line of the A matrix contains factors for the elements of vector X in the second constraint | STRUCT | [ 0, 0, 0, 0] |
| A3n.a31<br>A3n.a32<br>A3n.a33<br>A3n.a34 | 3. Line of the A matrix contains factors for the elements of vector X in the third constraint | STRUCT | [ 0, 0, 0, 0] |
| A4n.a41<br>A4n.a42<br>A4n.a43<br>A4n.a44 | 4. Line of the A matrix contains factors for the elements of vector X in the fourth constraint | STRUCT | [ 0, 0, 0, 0] |

| Parameter | Description | Type | Default |
|---|---|---|---|
| A5n.a51<br>A5n.a52<br>A5n.a53<br>A5n.a54 | 5. Line of the A matrix contains factors for the elements of vector X in the fifth constraint | STRUCT | [ 0, 0, 0, 0] |
| A6n.a61<br>A6n.a62<br>A6n.a63<br>A6n.a64 | 6. Line of the A matrix contains factors for the elements of vector X in the sixth constraint | STRUCT | [ 0, 0, 0, 0] |
| A7n.a71<br>A7n.a72<br>A7n.a73<br>A7n.a74 | 7. Line of the A matrix contains factors for the elements of vector X in the seventh constraint | STRUCT | [ 0, 0, 0, 0] |
| A8n.a81<br>A8n.a82<br>A8n.a83<br>A8n.a84 | 8. Line of the A matrix contains factors for the elements of vector X in the eighth constraint | STRUCT | [ 0, 0, 0, 0] |

* Values can be written back to these inputs during processing of the block by the block algorithm.

**Output parameters**

| Parameter | Description | Type | Default |
|---|---|---|---|
| Bad | 1= Status bad, optimization has failed | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see the error handling of LPOptim. | INT | 0 |
| J_Opt | Value of the performance criterion at the optimum | REAL | 0 |
| J_Act | Value of the performance criterion for the current assignment of the decision variables in accordance with input parameters X1...X4 | REAL | 0 |
| X1Opt | Optimum value of the 1st decision variable | REAL | 0 |
| X2Opt | Optimum value of the 2nd decision variable | REAL | 0 |
| X3Opt | Optimum value of the 3rd decision variable | REAL | 0 |
| X4Opt | Optimum value of the 4th decision variable | REAL | 0 |

## See also

LPOptim error handling (Page 1230)

### 6.14.7 LPOptim block diagram

**LPOptim block diagram**

A block diagram is not provided for this block.

# 6.15 MPC10x10 - Large predictive controller

## 6.15.1 Description of MPC10x10

### Object name (type + number) and family

Type + number: FB 1920

Family: Control

### Area of application for MPC10x10

The block is used for multivariable control, similar to the ModPreCon, typically as master controller in cascade structures.

Compared to the lean multivariable controller ModPreCon, the MPC10x10 has the following functional expansions:

- Greater capacity: up to 10x10 interacting manipulated and controlled variables (compared with ModPreCon: 4x4), up to 4 measurable disturbances for a dynamic disturbance compensation (feedforward) (ModPreCon: 1). Unused MV channels can be rededicated to disturbance variables, whereby the total number of manipulated variables and disturbances is limited to 14 (ModPreCon: 5).
- Dynamic online optimization considering constraints in each sampling step. The optimization problem is resolved without constraints with ModPreCon and the manipulated variables are limited afterward. This may result in solutions that are less than optimal if several constraints are active simultaneously in automatic mode.
- Processing of (low-priority) target values for the manipulated variables, which is mainly of interest for control tasks in which there are more manipulated variables than controlled variables or in which some controlled variables do not have to be kept at an exact setpoint.
- Integrated steady state operating point optimization even for control tasks in which the number of manipulated variables does not match the number of controlled variables.

### Method of operation and area of application

The block is used for Multivariable controller (Page 2888) dynamic processes. It can handle up to ten interacting manipulated and controlled variables as well as up to four measurable disturbances.

The MPC10x10 algorithm only works for stable processes with a step response that settles to a fixed value in a finite time. If the process is unstable in one of the main transfer functions or includes an integrator (level control, for example), the corresponding partial transfer function must be stabilized with a secondary controller.

A simple P controller (proportional component only) is sufficient as a slave controller for integrating processes. It is inserted between the manipulated variable output of MPC10x10 and the input of the unstable process section, and receives the output of the integrating process section as a manipulated variable. (Unstable data links are stabilized with this approach.)

You can find an explanation of "multivariable controls" and "non-phase-minimum response" in the Help on the Advanced Process Library > Definitions.

### Note on the area of application of the controller: Longer run times

Due to the principle on which it works, the runtime of MPC10x10 is significantly longer than that of ModPreCon and obviously much longer than that for PID controllers, because very large matrices have to be multiplied in the algorithm. The runtime is also determined by the number of the process and manipulated variables in the control algorithm. This is why the MPC10x10 is unsuitable for high-speed controls and is usually used for slow, complex control tasks in which controller sample times of more than 4 s are permitted.

The computing time required on the CPU is compensated for by the fact that very slow sample times of > 20 s are used for the typical MPC10x10 applications (see Advanced control templates). The MPC10x10 is then typically located in OB30 and can be interrupted by faster OBs. Some parts of the MPC10x10 algorithm have been outsourced to auxiliary blocks due to the limitations of the address space for SIMATIC function blocks and data blocks. The MPC10x10 uses two outsourced optimization methods:

- MPC_LP_OptiFB for operating point optimization which, in turn, needs an auxiliary block MPC_LPOptiFC .
- MPC_ActSetFB

as well as auxiliary blocks for matrices that are created by the MPC Configurator and can all be identified by the prefix MPC_* in their name.

There are different forms of implementation for the matrices:

- User DBs that are filled with data by one or more FCs during initialization because large matrices cannot be pre-assigned with values in the source code for the DB:
- Data block MPC_g with initialization FCs MPC_g_FC, MPC_gb...ge;
- Data blocks MPC_ESP, MPC_Hu_FB, MPC_Hinv_FB, MPC_Omega_FB and MPC_M_FB each with a dedicated initialization function MPC_Esp_FC... as well as
- a data block MPC_RgMVi (i=1..10) for each MV with initialization FCs MPC_RgMVia_FC and MPC_RgMVib_FC.
- User DBs filled with data by the MPC10x10 during runtime: MPC_Nn, MPC_Ms, MPC_MsOld.

As user, you only have to interconnect the main function block in the CFC and need not worry about the auxiliary blocks.

The static operating point optimization is called within the MPC10x10 block in the program section in which OB1 is executed. In cyclic operation OB3x, this avoids the additional computing time required by the optimizer, the acyclic time, i.e. which is only relevant for changes to the optimizer inputs.

### Operating principle

The MPC10x10 block is a model-based predictive multivariable controller. It uses a mathematical model of the process dynamics including all interactions as part of the controller. This model allows the process response to be predicted over a defined period in the future, also known as the prediction horizon.

Based on this prediction, a performance index

$$J = \left(\vec{w} - \vec{y}\right)^T R\left(\vec{w} - \vec{y}\right) + \Delta\vec{u}^T Q \Delta\vec{u} + \left(\vec{u}_{soll} - \vec{u}\right)^T S\left(\vec{u}_{soll} - \vec{u}\right)$$

is optimized (minimized) where the following applies:

- *w* contains the time series of the future setpoints within the prediction horizon,
- *y* contains the vector of the controlled variables in the future,
- *Δu* contains the future changes of the manipulated variables within the control horizon,
- $u_{soll}$ contains the target values for the manipulated variables *u*

The control horizon is generally significantly shorter than the prediction horizon. Changes of future manipulated variables are only planned within the control horizon.

If you increase the weighting in the Q diagonal matrix, the controller moves its manipulated variables more cautiously resulting in a slower but more robust control response. Using the weighting factors in the R diagonal matrix, you specify the relative significance of the individual controlled variables. A higher weighting (priority) for a controlled variable means that this CV moves more quickly towards the setpoint and remains more accurately at the setpoint in steady state if it is not possible to achieve all setpoints precisely.

The weightings for the target values of the manipulated variables in the S matrix are typically set much lower than the weightings of the controlled variables in the R matrix so that the controller must only handle the manipulated variable target values once it has reached all primary control aims regarding the controlled variables.

The algorithm is a variant of the DMC method (**D**ynamic **M**atrix **C**ontrol), comparable to the control algorithms in so-called "full-blown" MPC software packages that were previously linked to the process control system by external PCs.

The optimization problem is solved in ongoing automatic mode in each sampling step of the controller by initially neglecting the constraints. If a constraint violation is detected within the control horizon, an iterative optimization with the so-called "active set" method takes place. Only the currently active limitations are taken into consideration during each iteration step of the optimization. Manipulated variable limitations, both absolute and relating to the gradients, are treated in the algorithm of the function block as hard constraints that must not be violated. Precise setpoints or target zones for the controlled variables are taken into account during optimization as well as possible. The target zones for the controlled variables are therefore soft constraints, which are maintained as well as possible although they cannot be guaranteed. Using a reference variable filter for future setpoint trajectories, the control response of the controller can be fine-tuned during operation.

You can achieve significant improvements in control performance when individual disturbances can be measured, for example, variations in throughput. In this case, it is a good idea to take into account a model of the influence of this disturbance on the controlled variables when predicting the controlled variables so that the controller can react preemptively to such disturbances.

## Operating point optimization

The integrated steady state operating point optimization can be used when at least one controlled variable provides certain degrees of freedom. No exact setpoint is specified for such controlled variables. Instead there is a tolerance band, e.g.

- SP2OptHiLim...SP2OptLoLim

within which the process value, CV2, must remain. These areas can be defined for any subset of the relevant controlled variables. From an economic perspective, different values within the tolerance band can be more or less favorable. With the help of the optimization function, the optimal economic point can be found within the tolerance range. This is done by defining a target function (performance criterion), which depends on the manipulated variable and controlled variable of the predictive controller. This can be, for example, the economic yield of plant operation per time unit or it may involve specific costs or energy consumption.

## Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100) and in OB 1.

MPC10x10 and all its auxiliary blocks are provided in the block folder "MPC10x10 blocks" of the Advanced Process Library. After you have placed the block in the CFC, you still need to copy the fixed addressed global data blocks DB1 to DB22 into the project. This will make all auxiliary blocks of MPC10x10 available in the project, thereby making it executable and allowing the SCL source of the MPC configurator to be later compiled.

In contrast to the ModPreCon, only one instance of the MPC10x10 can be configured per SIMATIC CPU. The addresses of the MPC auxiliary blocks are determined during compilation of the main block by the SCL Compiler and cannot be modified afterwards. The addresses for the auxiliary blocks (DBs and FCs) used by the MPC10x10 are defined in the following symbol table. If you already have other blocks in the project assigned to the specified addresses, you need to move them.

| Icon | Address | Data type | Comment |
|---|---|---|---|
| MPC10x10 | FB 1920 | FB 1920 | Model Predictive Controller with online optim. |
| MPC_LP_OptiFB | FB 1922 | FB 1922 | Linear Optimization |
| MPC_ActSetFB | FB 1923 | FB 1923 | Modified Simplex Optimization with Active Set Problem Reduction for ModPreConL |
| MPC_LPOptiFC | FC 400 | FC 400 | Linear Optimization |
| MPC_g_FC | FC 401 | FC 401 | MPC Configurator |
| MPC_gb_FC | FC 402 | FC 402 | MPC Configurator |
| MPC_gc_FC | FC 403 | FC 403 | MPC Configurator |
| MPC_gd_FC | FC 404 | FC 404 | MPC Configurator |
| MPC_ge_FC | FC 405 | FC 405 | MPC Configurator |
| MPC_gz_FC | FC 406 | FC 406 | MPC Configurator |
| MPC_Hu_FC | FC 407 | FC 407 | MPC Configurator |
| MPC_Hinv_FC | FC 408 | FC 408 | MPC Configurator |
| MPC_Omega_FC | FC 409 | FC 409 | MPC Configurator |
| MPC_M_FC | FC 410 | FC 410 | MPC Configurator |

| | | | |
|---|---|---|---|
| MPC_RgMV1a_FC | FC 411 | FC 411 | MPC Configurator |
| MPC_RgMV1b_FC | FC 412 | FC 412 | MPC Configurator |
| MPC_RgMV2a_FC | FC 413 | FC 413 | MPC Configurator |
| MPC_RgMV2b_FC | FC 414 | FC 414 | MPC Configurator |
| MPC_RgMV3a_FC | FC 415 | FC 415 | MPC Configurator |
| MPC_RgMV3b_FC | FC 416 | FC 416 | MPC Configurator |
| MPC_RgMV4a_FC | FC 417 | FC 417 | MPC Configurator |
| MPC_RgMV4b_FC | FC 418 | FC 418 | MPC Configurator |
| MPC_RgMV5a_FC | FC 419 | FC 419 | MPC Configurator |
| MPC_RgMV5b_FC | FC 420 | FC 420 | MPC Configurator |
| MPC_RgMV6a_FC | FC 421 | FC 421 | MPC Configurator |
| MPC_RgMV6b_FC | FC 422 | FC 422 | MPC Configurator |
| MPC_RgMV7a_FC | FC 423 | FC 423 | MPC Configurator |
| MPC_RgMV7b_FC | FC 424 | FC 424 | MPC Configurator |
| MPC_RgMV8a_FC | FC 425 | FC 425 | MPC Configurator |
| MPC_RgMV8b_FC | FC 426 | FC 426 | MPC Configurator |
| MPC_RgMV9a_FC | FC 427 | FC 427 | MPC Configurator |
| MPC_RgMV9b_FC | FC 428 | FC 428 | MPC Configurator |
| MPC_RgMV10a_FC | FC 429 | FC 429 | MPC Configurator |
| MPC_RgMV10b_FC | FC 430 | FC 430 | MPC Configurator |
| MPC_Esp_FC | FC 432 | FC 432 | MPC Configurator |
| MPC_Ms | DB 1 | DB 1 | MPC Configurator |
| MPC_Nn | DB 2 | DB 2 | MPC Configurator |
| MPC_MsOld | DB 3 | DB 3 | MPC Configurator |
| MPC_g | DB 4 | DB 4 | MPC Configurator |
| MPC_gz | DB 5 | DB 5 | MPC Configurator |
| MPC_Hu | DB 6 | DB 6 | MPC Configurator |
| MPC_Hinv | DB 7 | DB 7 | MPC Configurator |
| MPC_Om | DB 8 | DB 8 | MPC Configurator |
| MPC_M | DB 9 | DB 9 | MPC Configurator |
| MPC_RgMV1 | DB 10 | DB 10 | MPC Configurator |
| MPC_RgMV2 | DB 11 | DB 11 | MPC Configurator |
| MPC_RgMV3 | DB 12 | DB 12 | MPC Configurator |
| MPC_RgMV4 | DB 13 | DB 13 | MPC Configurator |
| MPC_RgMV5 | DB 14 | DB 14 | MPC Configurator |
| MPC_RgMV6 | DB 15 | DB 15 | MPC Configurator |
| MPC_RgMV7 | DB 16 | DB 16 | MPC Configurator |
| MPC_RgMV8 | DB 17 | DB 17 | MPC Configurator |
| MPC_RgMV9 | DB 18 | DB 18 | MPC Configurator |
| MPC_RgMV10 | DB 19 | DB 19 | MPC Configurator |
| MPC_Esp | DB 20 | DB 20 | MPC Configurator |
| MPC_LP_Opti | DB 21 | FB 1922 | Instance DB FB 1922 |
| MPC_ActSet | DB 22 | FB 1923 | Instance DB FB 1923 |

After installation in CFC, follow the steps outlined below:

1. Excite the process with the controller in manual mode by applying a series of manipulated variable step changes. We recommend using the auxiliary function block, AutoExcitation, to generate excitation signals.
2. Record the measured data with the CFC trend display and export it to an archive file. You can also use a WinCC TagLogging TrendControl to record the data and to export the *.csv file. If the total number of manipulated and controlled variables exceeds the maximum number of trends in the CFC, you will have to use the WinCC TrendControl anyway.
3. Select the MPC10x10 instance in the CFC. Start the MPC Configurator under Edit > MPC Configurator.
   The working steps process identification, controller design and simulation of the closed control loop are executed in the MPC Configurator. You can find a detailed description of this procedure in the MPC Configurator help. You can find the help with a button in the MPC Configurator or directly under ...\Program Files (x86)\SIEMENS\STEP7\S7JMPC \s7jmpctb.chm.
4. Using the Configurator, create an SCL source code for the user data blocks (DBs) and other auxiliary blocks (FCs). The source contains all models and matrices required for an MPC10x10 instance.
5. Compile the SCL source with the S7-SCL compiler and archive the SCL source with the controller data.

Load the changes to the AS (with the option: Include user data blocks)

The values in the auxiliary blocks are applied in the controller by restarting the block by means of the input variable Restart:= true at the CFC block instance.

**Note**

During controller design in the MPC Configurator, a controller cycle time and an OB sampling time are calculated, displayed, and stored in the user data block. You yourself are responsible for ensuring that the MPC10x10 block is called in the cyclic interrupt level suitable for the OB sampling time. This is checked in the current MPC10x10 version during initialization. If the SampleTime parameter of the function block does not match the OB_SampleTime parameter of the user data block, a parameter assignment error (ErrorNum=3) is displayed. For controller cycle times greater than 5s, insert the MPC10x10 block in the OB30 and specify the appropriate cycle time for the OB30 in the hardware configuration of the SIMATIC CPU. Controller cycle times slower than 20 s cannot be set in the hardware configuration. The block would then be called every 20s and the slower sampling time automatically realized by an internal cycle reduction ratio in the block.

There are no templates for the MPC10x10 block for a process tag type in the Advanced Process Library because the controller is typically configured individually as master controller for several slave controllers.

### Startup characteristics

When the CPU starts up, the block always starts in manual mode. It is only possible to change to automatic mode when a user data block is loaded and the internal measured value memory in MPC10x10 is filled with data.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section I/Os of MPC10x10 I/Os (Page 1254)

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `AutAct.Value` |
| 6 | Invalid signal status |
| 7 | `ManAct.Value` |
| 8 - 9 | Not used |
| 10 | `MV1TrkOn.Value` AND NOT (`ManAct.Value` OR `OosAct.Value`) |
| 11 | `MV2TrkOn.Value` AND NOT (`ManAct.Value` OR `OosAct.Value`) |
| 12 | `MV3TrkOn.Value` AND NOT (`ManAct.Value` OR `OosAct.Value`) |
| 13 | `MV4TrkOn.Value` AND NOT (`ManAct.Value` OR `OosAct.Value`) |
| 14 | Internal error |
| 15 | `DB_Loaded` |
| 16 | `DV_Model Available` |
| 17 | OptimAct |
| 18 | `NOT(OptimAct)` |
| 19 | Not used |
| 20 | J_Mini |
| 21 | NOT(J_Mini) |
| 22 | ExciteOn AND ManAct.Value |
| 23 | `MV5TrkOn.Value AND NOT (ManAct.Value OR OosAct.Value)` |
| 24 | `MV6TrkOn.Value` AND NOT (`ManAct.Value OR OosAct.Value`) |
| 25 | `MV7TrkOn.Value AND NOT (ManAct.Value OR OosAct.Value)` |
| 26 | `MV8TrkOn.Value` AND NOT (`ManAct.Value OR OosAct.Value`) |
| 27 | `MV9TrkOn.Value AND NOT (ManAct.Value OR OosAct.Value)` |
| 28 | `MV10TrkOn.Value` AND NOT (`ManAct.Value OR OosAct.Value`) |
| 29 | Tracking mode; i.e. at least one of the original manipulated variables (without pseudo MVs, which are used as DVs) is tracked. |
| 30 | Not allocated |
| 31 | `Feature.Bit31`: Display of the predictions in the faceplate |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `MV1TargOn` |
| 1 | `MV2TargOn` |
| 2 | `MV3TargOn` |
| 3 | `MV4TargOn` |
| 4 | `MV5TargOn` |
| 5 | `MV6TargOn` |
| 6 | `MV7TargOn` |
| 7 | `MV8TargOn` |
| 8 | `MV9TargOn` |
| 9 | `MV10TargOn` |
| 10 - 30 | Not used |
| 31 | `MS_RelOp` |

## 6.15.2 MPC10x10 modes

### Operating modes of MPC10x10

The block can be operated using the following modes:

- Automatic mode (Page 79)
- Manual mode (Page 79)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

The aforementioned operating modes are valid for the block with all control channels (`MV1` ... `MV10`). Moreover, individual control channels can be tracked; see section MPC10x10 functions (Page 1243) for more information.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and bumpless switchover in the section Manual and automatic mode for control blocks (Page 79) .

**Note**

In contrast to PID controllers, it is permitted to run the MPC10x10 block in "automatic mode" without its actuating signals affecting the process because there is no risk of integrator windup.

### "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the section Manual and automatic mode for control blocks (Page 79) .

### "Out of service"

You can find general information about the "Out of service" mode in the Out of service (Page 77) .

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit5` =1 and `Feature.Bit5` =0 is as shown below:

| Output Parameters | `Feature.Bit5` =0 | `Feature.Bit5` =1 |
|---|---|---|
| SP1Out.ST | 16#80 | 16#60 |
| SP2Out.ST | 16#80 | 16#60 |
| SP3Out.ST | 16#80 | 16#60 |
| SP4Out.ST | 16#80 | 16#60 |
| SP5Out.ST | 16#80 | 16#60 |
| SP6Out.ST | 16#80 | 16#60 |
| SP7Out.ST | 16#80 | 16#60 |
| SP8Out.ST | 16#80 | 16#60 |
| SP9Out.ST | 16#80 | 16#60 |
| SP10Out.ST | 16#80 | 16#60 |
| SP1OpOut.ST | 16#80 | 16#60 |
| SP2OpOut.ST | 16#80 | 16#60 |
| SP3OpOut.ST | 16#80 | 16#60 |
| SP4OpOut.ST | 16#80 | 16#60 |
| SP5OpOut.ST | 16#80 | 16#60 |
| SP6OpOut.ST | 16#80 | 16#60 |
| SP7OpOut.ST | 16#80 | 16#60 |
| SP8OpOut.ST | 16#80 | 16#60 |
| SP9OpOut.ST | 16#80 | 16#60 |
| SP10OpOut.ST | 16#80 | 16#60 |
| CV1Out.ST | 16#80 | 16#60 |
| CV2Out.ST | 16#80 | 16#60 |
| CV3Out.ST | 16#80 | 16#60 |
| CV4Out.ST | 16#80 | 16#60 |
| CV5Out.ST | 16#80 | 16#60 |
| CV6Out.ST | 16#80 | 16#60 |
| CV7Out.ST | 16#80 | 16#60 |
| CV8Out.ST | 16#80 | 16#60 |
| CV9Out.ST | 16#80 | 16#60 |
| CV10Out.ST | 16#80 | 16#60 |

## 6.15.3 MPC10x10 functions

### Functions of MPC10x10

The functions for this block are listed below.

### Generating and limiting the manipulated variable

The manipulated variable MV1 ... MV4 (hereinafter referred to as MVx, x = 1 ... 10) can be generated as follows:

| ManAct | MVxTrkOn | MVx | Limit monitoring | State |
|---|---|---|---|---|
| 1 | - | Manx | ManxHiLim<br>ManxLoLim | "Manual mode", set by the operator |
| 0 | 1 | MVxTrk | MVxHiLim<br>MVxLoLim | Tracking with limitation |
| 0 | 0 | Automatic manipulated variable | MVxHiLim<br>MVxLoLim | "Automatic mode": Predictive controller algorithm |

**Remark**

If the controller is in "Out of service" mode, the output parameters MV1 ... MV10, depending on the Feature Bit (Neutral position manipulated variable takes effect at startup (Page 176)), are set to the last valid value in manual mode or the corresponding neutral position manipulated variable (SafePos1 ... SafePos10). Refer to the Out of service (Page 77) section for more on this.

The limited operating range (between MVxHiLim and MVxLoLim) is typically smaller in automatic mode than in manual mode. With regard to the limited range of validity of a linear process model for approximating a non-linear process response, this allows the stability of the closed control loop to be guaranteed within the control range in automatic mode.

The gradients of the manipulated variable (changes per second) are limited to MV1RaLim to MV10RaLim in "automatic mode". Gradient limitation applies both to the positive and negative directions.

### Tracking and limiting a manipulated variable

The block provides the standard function Tracking and limiting a manipulated variable (Page 222).

In contrast to PID controllers, tracking the manipulated variables (MV1 ... MV10) is enabled for specific channels via one of the input parameters MV1TrkOn ... MV10TrkOn. The corresponding manipulated variable is then tracked by the interconnectable input parameters MV1Trk ... MV10Trk.

### Setting the setpoint internally

With this block, the setpoint must always be set internally at the I/Os SP1 ... SP10. These are normally operated in the faceplate. In special situations, you can interconnect the setpoints but they can then no longer be changed using the faceplate.

### Setpoint tracking in manual mode

In this situation (`SP_TrackCV` = 1), the internal setpoints `SP1` ... `SP10` are tracked to the assigned process values `CV1` ... `CV10` in "manual mode". This function allows a bumpless transfer to "automatic mode". After the transfer, the setpoints can be changed by the operator again.

### Setpoint filters

The setpoint filter is the only way of changing the action of the predictive controller without having to create a new user data block with the MPC Configurator and reinitialize the controller. The specified time constant `PreFilt1` ... `PreFilt10` of the setpoint filter can be interpreted as the required settling time of this `CV` channel following a setpoint step change. As the time constant setting increases, the controller works more slowly and less aggressively. In particular, this reduces the influence of a setpoint step change in one control channel on neighboring control channels.

Internally, the MPC10x10 block works with trajectories of future setpoints that are compared with the predicted movements of the controlled variables. Without the setpoint filter, it is assumed that the current setpoint will continue to remain valid in the future within the prediction horizon. If there is a setpoint step change, this means that the full value of the new setpoint will be required in the near future although the process cannot achieve this (according to the prediction). With the setpoint filter, an asymptotic setpoint trajectory (first order) is calculated from the current process value to the required setpoint so that the required setpoint is reached in the specified time.

**Note**

The setpoint filter also comes into effect without a setpoint step if the process value deviates significantly from the setpoint due to disturbances. This means that the filter not only slows down the control response but indirectly also the response to disturbances.

The control response can only be slowed down by the setpoint filter and not accelerated; when the value is 0, the prefilter is deactivated. It is therefore advisable to set the basic controller tuning in the MPC Configurator with the "Manipulated variable move penalty" parameter and then to optimize this inside the software using the function for simulation of the closed control loop. The setpoint filter should then only be used for fine tuning of the controller action in the real-world process.

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Controlled variable (`SimCVx` , `SimCVxLi`)

### Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 240).

### Error signal generation and dead band

The block provides the standard function Error signal generation and dead band (Page 217).

In the predictive controller, the error signal is generated over the entire prediction horizon for each control channel as the deviation between the predicted movement of the process (starting at the current process value CV1 ... CV4) and future setpoint settings (ending at SP1 ... SP10) and used to calculate the manipulated variable.

In principle, the effect of the dead bands SP1DeadBand to SP10Deadband is the same as in a PID controller, but extends over the entire future prediction horizon. In other words, if, for example, the predicted controlled variable CV1 in the entire prediction horizon is within the band SP1 ± SP1DeadBand, the controller sees no reason whatsoever to change any manipulated variable. These are therefore also known as CV bands. In contrast to the manipulated variable limits, these are not hard constraints that need to be adhered to at all costs.

In multivariable controllers, it is advisable to make use of the fact that from the perspective of the application only some of the controlled variables need to move to a specified setpoint exactly while others only need to remain within a defined range.

A typical example would be product quality characteristics for which a tolerance band is specified. While a dead band in a PID controller tends to put stability at risk, CV bands in individual controller channels generally relieve the multivariable controller overall.

Using CV bands, the action of a soft override control can be achieved.

### Use case for error signal generation with dead band

As long as the pressure in a reactor remains within the specified safety limits, the controller is interested only in product quality. However, as soon as the pressure threatens to leave the permitted range (in other words, inside the prediction horizon it moves towards an illegal value in the future), the pressure control cuts in. By weighting the controlled variables in the performance index (see MPC Configurator), the user can specify that predicted violations of the pressure limits are given a particularly high weighting.

### Predictive controller algorithm

The MPC10x10 block is derived from the familiar DMC algorithm (**D**ynamic **M**atrix **C**ontrol) .

Future changes to the manipulated variable within the control horizon are calculated from future predictive control deviations. To accomplish this, first the ideal solution is determined in each sampling step using an analytical formula without consideration of constraints. If the ideal solution comes into conflict with a constraint, an iterative optimization process is used to find a solution of the dynamic quadratic optimization problem with constraints.

Based on the principle of the receding horizon, only the first value is taken from the calculated vector of the optimum manipulated variable moves over the entire control horizon and applied to the process. In the next step, the newly arrived process values are taken into account and the calculation repeated over the entire prediction horizon.

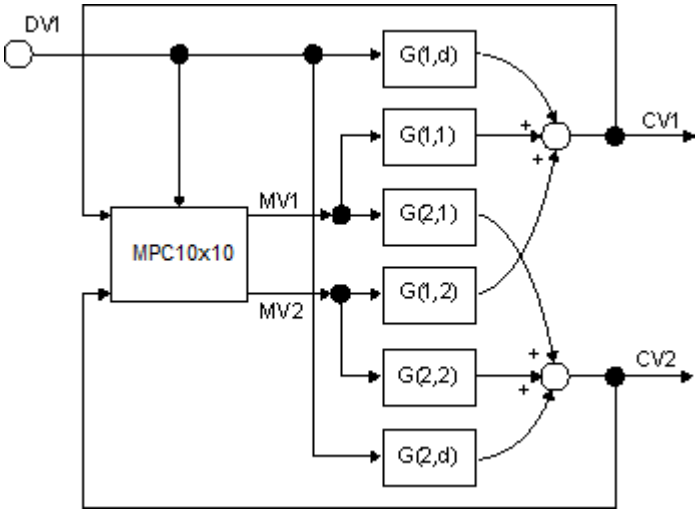
With predictive controllers, the manipulated variable moves are based on the control deviations predicted in the future, while with a PID controller, they are based on (typically integrated) error signals of the past. This can be interpreted as a "looking ahead" strategy.

### Anti-windup

When manipulated variable limits are active, anti-windup measures are taken automatically within the controller. The prediction equations use the real limited values of the manipulated variables instead of theoretically calculated values.

### Model-based disturbance compensation

Model-based disturbance compensation can and should be used if one or more known disturbances have a strong influence on the process and their causes can be measured.




The effects of a measurable disturbance (`DV1..4` I/O) on all controlled variables `CV1 ... CV10` can be estimated when the controller is taken into manual mode. This means that no movements of the controlled variables whatsoever result from changes to the manipulated variables and all movements result from the disturbance. If the disturbance (e.g. ambient temperature) can be measured but cannot be actively adjusted, it may be necessary to search through a data archive to find the time segments in which the disturbance changed.

The identification of the transfer functions from disturbances `DV1...DV4` to all controlled variables `CV1 ... CV10` (disturbance models G(1,d) and G(2,d) in the graphic above) is performed with the MPC Configurator and is analogous to the identification of the main transfer functions (G(1,1) to G(2,2)). The measured disturbances are then linked to the `DV1...DV4` inputs of the MPC10x10 block and disturbance compensation is activated with `DV_On = 1`. As a result, the effect of the measurable disturbances is taken into account in the prediction and the controller can start counter measures in advance before the disturbance has a massive influence on the controlled variables.

Such disturbance compensation is especially effective when the disturbance is constant at times and changes from time to time. If a disturbance fluctuates constantly or oscillates, however, the feedforward control should not be enabled during operation of the controller to avoid constant oscillation of the manipulated variables, although it should be taken into account in the MPC Configurator when creating the process model.

If there is no disturbance model in the user data block, the signals at the `DV1...DV4` input are ignored.

Typical examples of measurable disturbances are inlet flows in distillation columns or throughput of continuous reactors.

### Predictive controller with more than four measurable disturbances

If you want to plan for more than four measurable disturbances in an application, but do not need all ten manipulated variable channels of the MPC10x10 block, you can dedicate the first of the previously unused MV channels for disturbance feedforwarding.

Example: You only have two control actions available, so only use MV1 and MV2. Then connect the additional measurable disturbance DV5 to the MV3Trk input parameter and set MV3TrkOn = 1. For recording of training data for the predictive controller, declare MV3Trk as the third disturbance and also use the CFC trend recorder to record the effect of changes to DV5 on all controlled variables.

Use the MPC Configurator to then determine a process model that describes the effects of DV5. However, if the disturbance variable DV5 in "automatic mode" of the controller fluctuates due to external influences, the effect of such changes is taken into consideration for predicting future process reactions through MV3Trk and predictive methods can be used to compensate this disturbance. The performance of the disturbance compensation is exactly the same as that for regular feedforward control via the input parameters DV1...DV4 and DV_On = 1.

If you want to disable this disturbance compensation in runtime with the unused MV5 manipulated variable, you need to insert a SelA02In selector block in front of the MV3Trk input. This allows you to set a constant zero for MV3Trk instead of the measured value DV5, which stops the effect of MV3 on prediction. (Due to this reassignment, MV3TrkOn must always remain 1 to prevent the controller from changing the value of MV3.)

This way, additional measurable disturbances can be selected. However, the sum of the manipulated variables and disturbances may not exceed a total number of 14.

At the output variables NumberOrigMVs, the number of original manipulated variables is displayed that can actively be influenced by the controller. The other manipulated variables (in the above example MV3) are only used for disturbance feedforward and are therefore always in tracking mode. They are referred to as pseudo MVs.

### Control of square and non-square systems

In multivariable controllers, the number of manipulated variables should ideally be the same as the number of controlled variables. This is known as a "square system". As long as constraints do not influence operation, the controller can, in principle, track all controlled variables exactly to the selected setpoints.

If there are less manipulated variables than controlled variables or if individual manipulated variables have reached their limits, degrees of freedom are lacking in the control problem. This means that it is not possible for all setpoints to be reached exactly.

The MPC10x10 algorithm then finds a compromise that can be influenced by the selection of controlled variable weightings (priorities) in the MPC Configurator: Controlled variables with higher priority will have lower control deviations.

If there are more manipulated variables than controlled variables or if some of the controlled variables are already within their setpoint bands, there are surplus degrees of freedom in the control problem.

The MPC10x10 algorithm can use these degrees of freedom to move the manipulated variable in the direction of the specified target value, whereby the weighting of the manipulated variable target value in the performance index is typically substantially less than the weighting of setpoints for the controlled variables. If the manipulated variables result in costs (e.g. mass flows of steam or other forms of energy consumption), the manipulated variable target values

can be set somewhat lower than the values previously observed at the operating point. The controller then attempts to reduce the manipulated variables (the costs) in the direction of the target values as long as the achievement of the primary control target provides some degree of freedom. In general, the advantages of the dynamic online optimization in the MPC10x10 are especially attractive for non-quadratic systems, because the restrictions of the lean algorithm of ModPreCon are overcome.

## Control of linear and non-linear systems

The MPC10x10 algorithm is based on a linear, time invariant process model. As a result, in much the same way as a PID controller, it is suitable above all for controlling non-linear systems around a fixed operating point.

Again analogous to the PID controller, there are, however, several possibilities with which the area of application can be extended in context of non-linear systems:

**Compensation functions between controller and controlled system:**

It is, for example, possible to compensate the effect of a non-linear valve characteristic curve using a polygon block between the `MV` output and the control input of the valve block. Care must be taken when implementing the manipulated variable limits. In the same way, the effect of a non-linearity at the output of the controlled system (for example a sensor characteristic curve) can be compensated by a polygon block before the `CV` input of the controller. Remember that the corresponding SP must also be equivalently transformed. In both cases, the compensation functions become part of the controlled system from the perspective of the controller. The aim is always to keep the overall response of the controlled system consisting of process and compensation elements as linear as possible.

**Trajectory control:**

This approach neatly combines the advantages of an open loop controller (Feedforward Control) with those of a closed loop controller with process value feedback (Closed Loop Control). The controller follows a previously optimized trajectory of setpoints and manipulated variables; in other words, it only needs to compensate small deviations between the stored trajectory and the current plant state. A trajectory is an optimum series of manipulated variables over time and the process values that match them. The required manipulated variables are read by means of the `MV10Traj` inputs into the MPC10x10 block and added to the values of the manipulated variable calculated by the algorithm (in automatic mode only). Among other things, the advantage of this is that the effective manipulated variable acting on the process as the sum of the trajectory and controller action is limited correctly as configured. The process values from the trajectory are linked to the corresponding setpoint specifications `SP1` ... `SP10` of the controller. As long as the process reacts exactly as planned in the trajectory, it will respond to the series of manipulated variables from the trajectory with the corresponding series of process values and the control deviation is zero. It is generally known that a non-linear dynamic process can linearized around a fixed operating point or a steady state of the system. It is also possible to linearize it around a trajectory.

## Forming the signal status for blocks

The block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `DV1.ST`
- `DV4.ST`
- `CV1.ST`
- `CV2.ST`
- ...
- `CV10.ST`

### Configurable reactions using the `Feature` parameter

An overview of all the reactions that are provided by the `Feature` parameter is available in the section Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 4 | Setting switch or button mode (Page 177) |
| 5 | Changes signal status of outputs in OOS (Page 208) |
| 8 | Delete history of MV and DV moves during restart (Page 201) |
| 15 | Neutral position manipulated variable takes effect with "out of service" operating mode (Page 177) |
| 16 | Neutral position manipulated variable takes effect at startup (Page 176) |
| 24 | Enabling local operator authorization (Page 169) |

### Operator permissions

The block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" |
| 1 | Not used |
| 2 | 1 = Operator can switch to "Out of service" mode |
| 3 | Not used |
| 4 | Not used |
| 5 | 1 = Operator can change the setpoint 1 |
| 6 | 1 = Operator can change the manipulated variables of all channels |
| 7 | 1 = Operator can change operating high limits of the setpoints for all channels |
| 8 | 1 = Operator can change operating low limits of the setpoints for all channels |
| 9 | 1 = Operator can change the setpoint 2 |
| 10 | 1 = Operator can change the setpoint 3 |
| 11 | 1 = Operator can change the setpoint 4 |

| Bit | Function |
|---|---|
| 12 | 1 = Operator can change the setpoint filter of all channels |
| 13 - 16 | 1 = Operator can change the setpoint 5...8 |
| 17 | 1 = Operator can enable the track setpoint in "manual mode" function |
| 18 | 1 = Operator can activate the model-based disturbance compensation function |
| 19 | 1 = Operator can change the setpoint 9 |
| 20 | 1 = Operator can change the setpoint 10 |
| 21 | 1 = Operator can only activate the "Prediction Mode" function |
| 22 | 1 = Operator can change the target values of all channels |
| 23 | 1 = Operator can change the dead band parameter of all channels |
| 24 | 1 = Operator can change all parameters of the operating point optimization |
| 25 | 1 = Operator can change the simulation value `SimCV1..10` |
| 26 | 1 = Operator can activate the Simulation function |
| 27 | 1 = Operator can activate the Release for maintenance function |
| 28 | 1 = Operator can change the manipulated variable limits of all channels |
| 29 | 1 = Operator can change the gradient limits of manipulated variables of all channels |
| 30 | 1 = Operator can activate the operating point optimization |
| 31 | 1 = Operator can deactivate the operating point optimization |

The block has the following permissions for the `OS1Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Operator can activate/deactivate the `MV target` values of all channels |
| 1-15 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

## Release for maintenance

The block provides the standard function Release for maintenance (Page 70).

## Specifying the display area for process and setpoint values as well as operations

This block provides the standard function Display and operator input area for process values and setpoints (Page 233).

In contrast to PID controllers, there are no separate parameters for bar limits. The setpoints limits are used for all setpoint and actual value bars; manual limits are used as bar limits for all manipulated variable bars.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Integrated static operating point optimization

The integrated steady state operating point optimization can be used if no exact `SPi` setpoint is specified for at least one controlled variable (index i =1...10), but rather the `CVi` process value must remain within a tolerance range `SPiOptHiLim`...`SPiOptLoLim`. The tolerance range must, of course, be within the valid setpoint limits `SPiHiLim`...`SPiLoLim` for this control channel. The tolerance range `SPiOptHiLim`...`SPiOptLoLim` is not coupled to the operator-controlled setpoint `SPi`. If the setpoint is changed, the tolerance range is not automatically shifted. If you want to do this nevertheless, interconnect the `SPiOpOut` output parameters via two adders using the width of the tolerance range to the inputs `SPiOptHiLim` and `SPiOptLoLim`.

From an economic perspective, different values within the tolerance range can be more or less favorable. With the help of the optimization function, the optimal economic point can be found within the tolerance range.

This is done by defining a target function (performance criterion), which depends on the manipulated variable and controlled variable of the predictive controller. This can be, for example, the economic yield of plant operation per time unit or it may involve specific costs or energy consumption.

```
J= GradMV1*MV1 +GradMV2*MV2 +... +GradMV10*MV10
+GradCV1*CV1 +GradCV2*CV2 +...+GradCV10*CV10
+J0
```

You specify the individual `GradXVi` coefficients of the gradient vector as input variables at the MPC10x10 function block in the CFC or in the parameter view of the faceplate. If individual coefficients vary with time, e.g. they are dependent on current market prices, you can also interconnect these input variables. If individual manipulated or controlled variables have no influence on the performance criterion, leave the corresponding coefficients at the default value, zero.

You can use the binary `J_Mini` input parameter to specify whether the target function is to be maximized or minimized, based on whether this involves yields or costs (`J_Mini = 1` : minimization).

The term `J0` combines all contributions to the target function that do not depend on manipulated variables and controlled variables. These contributions have no effect on the optimum values of the decision variables, but are applied for calculating the current value of specified performance index according to the above-mentioned formula.

Within the controller, the terms of the target function that depend on controlled variables are converted to make their dependence on the manipulated variables visible. To do this, the steady state process model from the MPC Configurator is used.

Constraints for the controlled variables take the form of the above-mentioned tolerance ranges for setpoints. The controller takes care of adhering to the manipulated variable constraints, in any case; they do not have to be separately specified as constraints for the optimization.

Enable the optimization using the binary input variable `OptimizeOn` in the controller faceplate. The optimizer then returns setpoints within the tolerance ranges that are optimal with respect to the performance index. These setpoints are then sent to the control algorithm, which handles them in the same way as conventionally specified setpoints (with or without dead band). The operable `SP1...SP10` setpoints are not tracked to the optimized setpoints; when optimization is disabled, the old setpoints from the faceplate take effect once again. When selecting variables for archiving and graphic plotter, ensure you use the `SP1Out...SP10Out` setpoint actually in effect and not the `SP1...SP10` input variables.

The current value of the performance index is displayed at the `J_Actual` output variable.

You can find additional information about the topic of steady state operating point optimization in the online help for the MPC configurator.

### Display of the prediction of free movement

The prediction of free motion ("future without control") is a forecast for the future behavior of the process within the overall prediction horizon, under the assumption that all manipulated variables are frozen at their current values. The time length of the prediction horizon is indicated in the output parameter `PrediHorizon` in the [s] unit.

The prediction of free motion is recalculated in each sampling step within the control algorithm. If the manipulated variables are at constant values in manual mode, the prediction of the free movement is actually a realistic prediction for the future process response. It can therefore be represented graphically in the faceplate at least in terms of quality. For this purpose, five equidistant interpolation values are copied from the prediction horizon, and displayed in the standard view of the faceplate as a vertical bar next to the current process value.

Example: The prediction horizon is 1800s=30min and the current time is labeled with the index k. The prediction for k+6min, and next to it k+12min, up to k+30min then appears on the right next to the bar of the current process value. If the upper borders of each bar are virtually connected with a line (red in the picture), you can imaging the curve created by the future course of the process value over the next half hour.

In automatic mode (tracking function is switched off), the value of the manipulated variables fluctuates with each sampling step. The prediction of free movement is then only a fictional mathematical formulation within the algorithm, and not a realistic prediction for the future process response. This is why the prediction is only displayed in manual mode or in automatic mode with active tracking function. The display can be generally suppressed using `Feature.Bit31`.

### Prediction without control response

In this special "operating mode" (comparable to the block-internal simulation), the controller only monitors the process and indicates what it would like to do in the next sampling step without actively intervening in the process. This allows you as the user to build trust before "switching active" the controller the first time, i.e. actively intervening in the process.

"Prediction Mode" is activated via the binary input variable `PredictMode` or in "Parameters" in the faceplate view. Setpoints and process values are read as in normal automatic mode. The prediction of the free movement and manipulated variable change for the next sampling step

are calculated as in normal automatic mode. The starting point for the prediction of the manipulated variable for the next sampling step, however, is the current process value of the follow-up control loop at the MV1Trk...MV10Trk tracking inputs. The predicted manipulated variables are not output at the normal outputs MV1...10 but rather at the MV1Pred...MV10Pred outputs, which were especially introduced for this purpose, and are displayed in the standard view of the faceplate on the left next to MV1...10 as long as "Prediction Mode" is active.

- When the controller is in automatic mode, in "Prediction Mode" all MVi (i=1..10) are set to match the assigned MViTrk input parameters, similar to tracking mode.
- When the controller is in manual mode, all MVi are set to the desired manual values regardless of "Prediction Mode".
- When "Prediction Mode" is disabled, all MViPred always equal the respective MVi.

### Automatic process excitation for model identification

In order to determine the process model for the model predictive controller, the process must be artificially excited in order to observe its dynamic response and record it in the form of training data. This excitation signal can be specified manually in manual mode of the controller.

Alternatively, a suitable signal can be generated automatically in the form a defined, symmetrical sequence of manipulated variable steps. The trigger signals are calculated by an auxiliary function block, "AutoExcitation", which is built into the process tag type and interconnected to MPC10x10 .

Additional MV1Excite...MV10Excite input variables are required for this on the controller function block. Process excitation is performed in manual mode of the controller, because automatic mode cannot be activated before modeling. The new "Operating mode" process excitation can only be controlled via the ExciteOn input bit on the engineering system, and not on the operator station, because the CFC is needed for configuration of the AutoExcitation block. However, process excitation must be displayed on the OS in standard view at the lower left.

Manual intervention via faceplate remains possible even during process excitation. The values of the MV1Excite...MV10Excite input parameters are therefore only written event-based to the MV1Man...MV10Man values that is only if they change.

You can find additional details on automatic process excitation in the online help for the MPC configurator.

### See also

Predictive control of a non-linear process (ModPreConNonLinSim) (Page 2879)

## 6.15.4 MPC10x10 error handling

### Error handling of MPC10x10

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` output parameter can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Default value when the block is installed; this message is irrelevant. |
| 0 | There is no error. |
| 3 | Configuration error: Block is inserted in the wrong OB, cycle time of the OB is not suited for controller configuration. |
| 31 | The value of `CV1` can no longer be displayed in the real number field or is not a number. |
| 32 | The value of `CV2` can no longer be displayed in the real number field or is not a number. |
| ... | .. |
| 40 | The value of `CV10` can no longer be displayed in the real number field or is not a number. |
| 41 | The `MV_Trk1` value can no longer be displayed in the real number field or is not a number. |
| 42 | The `MV_Trk2` value can no longer be displayed in the real number field or is not a number. |
| ... | ... |
| 50 | The `MV_Trk10` value can no longer be displayed in the real number field or is not a number. |
| 51 | `AutModLi` = 1 and `ManModLi` = 1 |
| 90 | The controller matrix could not be loaded from the user data block. |

The ErrorOpt output parameter is used to output the status of the lower-level LPOptim block.

## 6.15.5 MPC10x10 messaging

### Messaging

This block does not offer messaging.

## 6.15.6 MPC10x10 I/Os

I/Os of MPC10x10

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

## Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutModLi | 1 = Automatic mode via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutModOp | 1 = Automatic mode via operator (controlled by ModLiOp = 0) | BOOL | 0 |
| BatchEn | 1 = Enabled for allocation by batch control | BOOL | 0 |
| BatchID | Batch number | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| CV1...CV10 | Controlled variable 1...10 (process value) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| CV1_Unit...CV10_Unit | Unit of measure for controlled variable 1...10 | INT | 1001 |
| DV_On | 1 = Activate the disturbance variable feedforward using DV | BOOL | 1 |
| DV1...DV4 | Disturbance variable 1...4 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| DV1_Unit...DV4_Unit | Unit of measure for DV1...DV4 | INT | 1342 |
| DV1DeadBand...DV4DeadBand | Dead band for DV1...DV4 | REAL | 1.0 |
| DynOptOn | 1 = Activate dynamic optimization with constraints.<br>0 = Ignore constraints for optimization. | BOOL | 1 |
| ExciteOn | 1 = Automatic process trigger; MViExite input parameters are written to the MVi outputs | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Feature | I/O for additional functions (Page 1243) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 1 |
| GradCV1... GradCV10 | Gradient vector for performance criterion, element (factor) for CV1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| GradMV1... GradMV10 | Gradient vector for performance criterion, element (factor) for MV1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| J_Actual_Unit | Physical unit of performance J_Actual | INT | 0 |
| J_Mini | 1 = minimize J,<br>0= maximize J | BOOL | 0 |
| J0 | Value of the performance criterion in the operating point | REAL | 0.0 |
| ManModLi | 1 = Manual mode via interconnection or SFC (effective if ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManModOp | 1 = Manual mode via OS operator (effective if ModLiOp = 0) | BOOL | 0 |
| ModLiOp | Operating mode switchover by:<br>1 = Interconnection or SFC<br>0 = Operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_RelOp | Operator input for release for maintenance<br>1 = MS release requested | BOOL | 0 |
| MV1_Unit...MV10_Unit | Unit of measure for manipulated variable 1...10 | INT | 1342 |
| MV1Excite...MV10Excite | Default value for MV1..10 during the automatic process trigger for model generation | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV1HiLi | High limit of manipulated variable MV1 | REAL | 100.0 |
| MV1LoLi | Low limit of manipulated variable MV1 | REAL | 0.0 |
| MV1Man | Manual value: Operator input for setting the manipulated variable MV1 in manual mode | REAL | 0.0 |
| MV1ManHiLim | High limit of manipulated variable MV1 in manual mode | REAL | 100.0 |
| MV1ManLoLim | Low limit of manipulated variable MV1 in manual mode | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MV1RaLim | Gradient limiting of the manipulated variable MV1 [MV1_Unit/ second] | REAL | 100.0 |
| MV1TargOn...MV10TargOn | 1 = Activate MV1Target | BOOL | 1 |
| MV1Trk | Tracking value for the manipulated variable MV1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV1Traj | Trajectory value that is added to the manipulated variable MV1 | REAL | 0.0 |
| MV1TrkOn...MV10TrkOn | 1 = Tracking of manipulated variable MV1...MV10 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = Allocated by SIMATIC BATCH | BOOL | 0 |
| OosLi | 1 = Out of service, via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp | 1 = Out of service, via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, to be connected with the Out output parameter of the upstream block OpStations | DWORD | 16#00000000 |
| OptimOffOp | 1 = Disable operating point optimization, normal setpoints SP1...SP10 in effect | BOOL | 0 |
| OptimOnOp | 1 = Enable operating point optimization, optimized setpoints SP1Out...SP10Out in effect | BOOL | 0 |
| OS_Perm | I/O for operating permissions (Page 1243) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| PredictMode | 1 = "Prediction only mode" enabled, prediction only, no intervention in the process | BOOL | 0 |
| PreFilt1...PreFilt10 | Settling time [s] of the setpoint filter for setpoint SP1...SP10 | REAL | 0.0 |
| Restart | 1 = Restart of the block and adoption of the data from the user block of the MPC Configurator | BOOL | 0 |
| SafePos1...SafePos10 | 1 = Safety setting for MV1...MV10 is Man1HiLim...Man10HiLim,<br>0 = Safety setting for MV1...MV10 is Man1LoLim...Man1LoLim, | BOOL | 0 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 1.0 |
| SelFp1 | 1 = Call a block saved in this parameter as an additional faceplate in the standard view | ANY | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SelFp2 | 1 = Call a block saved in this parameter as an additional faceplate in the preview | ANY | |
| SimCV1... SimCV10 | Simulation value for control variable CV1...CV10, which is used for SimOn = 1 | REAL | 0.0 |
| SimCV1Li... SimCV10Li | Simulation value for control variable CV1...CV10, which is used for SimOnLi.Value = 1 (and SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SimLiOp | Activation/deactivation of the simulation by:<br>1 = Interconnection or SFC<br>0 = Operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn | 1 = Enable simulation | BOOL | 0 |
| SimOnLi | 1 = Enable simulation via interconnection or SFC (if SimLiOp = 1)) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_TrkCV | 1 = Setpoints follow the CVs in manual mode and in tracking | BOOL | 0 |
| SP1...SP10 | Setpoint 1...10 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| SP1DeadBand... SP10DeadBand | Width (radius) of the dead band control of CV1...CV10 | REAL | 0.0 |
| SP1HiLim...SP10HiLim | Setpoint high limit 1...10 | REAL | 100.0 |
| SP1LoLim...SP10LoLim | Setpoint low limit 1...10 | REAL | 0.0 |
| SP1OptHiLim... SP10OptHiLim | High limit for optimization of setpoint 1...10 | REAL | 0.0 |
| SP1OptLoLim... SP10OptLoLim | Low limit for optimization of setpoint 1...10 | REAL | 0.0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutAct | 1 = Automatic mode is active<br>0 = Manual mode is active<br>(applies to all manipulated variable channels) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CV1Out...CV10Out | Copy CV1...CV10, preferably as output parameter | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| ErrorEndless | Error: infinite iteration loop in the dynamic optimization | BOOL | 0 |
| ErrorNum | Output of pending error number. You can find information about the error numbers that are output by this block at MPC10x10 error handling (Page 1253) | INT | -1 |
| ErrorOpt | Error number of the integrated operating point optimization function, see MPC10x10 error handling (Page 1253) | INT | 0 |
| Fut1_y1... Fut10_y5 | Prediction of free movement of CV1...CV10 for five future points in time within the prediction horizon | REAL | 0.0 |
| J_Actual | Current value of the performance criterion | REAL | 0.0 |
| Loop1Closed... Loop10Closed | 1 = MV1 is used by the algorithm for closed loop control, i.e. AutAct AND NOT MV1TrkOn<br>0 = Control loop for MV1 open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManAct | 1 = Manual mode enabled, for all control channels | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Release | Release for maintenance:<br>1 = Release by OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV1...MV10 | Manipulated variable 1..10 (controller output signal) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| MV1HiAct | 1 = High limit of manipulated variable 1 reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV1LoAct | 1 = Low limit of manipulated variable 1 reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV1Pred | One-step prediction for MV1 in the "Prediction without control response" mode | REAL | 0.0 |
| MV1TargOut...MV10TargOut | Target value output (optimum value) for manipulated variable MV1...MV10 | REAL | 0.0 |
| NumberCVs | Number of control variables used | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| NumberDVs | Number of disturbances used | INT | 0 |
| NumberMVs | Number of manipulated variables used | INT | 10 |
| NumberOrigMVs | Number of original manipulated variables that are actively influenced by the controller (without pseudo MVs that are used as DVs | INT | 0 |
| OosAct | 1 = Block is "out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feed-forwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OptimAct | 1 = Operating point optimization enabled, setpoints calculated by the optimization are used<br>0 = Optimization off | BOOL | 0 |
| OS1PermLog | Display of OS_Perm with settings changed by the block algorithm | WORD | 16#FFFF |
| OS1PermOut | Display of OS_Perm | WORD | 16#FFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| PrediHorizon | Prediction horizon [s] | REAL | 0.0 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP1OpOut...SP10OpOut | Copy of the operable setpoint 1 for step-enabling | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| SP1Out...SP10Out | Setpoint 1 actually used by controller | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 1234) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1234) | DWORD | 16#00000000 |

## 6.15.7 MPC10x10 block diagram

### MPC10x10 block diagram

A block diagram is not provided for this block.

## 6.15.8 Operator control and monitoring

### 6.15.8.1 MPC10x10 views

#### Views of the MPC10x10 block

The block MPC10x10 provides the following views:

- MPC10x10 standard view (Page 1261)
- Trend view (Page 341)
- MPC10x10 parameter view (Page 1267)
- MPC10x10 CV parameter view (Page 1269)
- MPC10x10 MV parameter view (Page 1270)
- MPC10x10 preview (Page 1271)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for MPC10x10 (Page 1274)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

#### See also

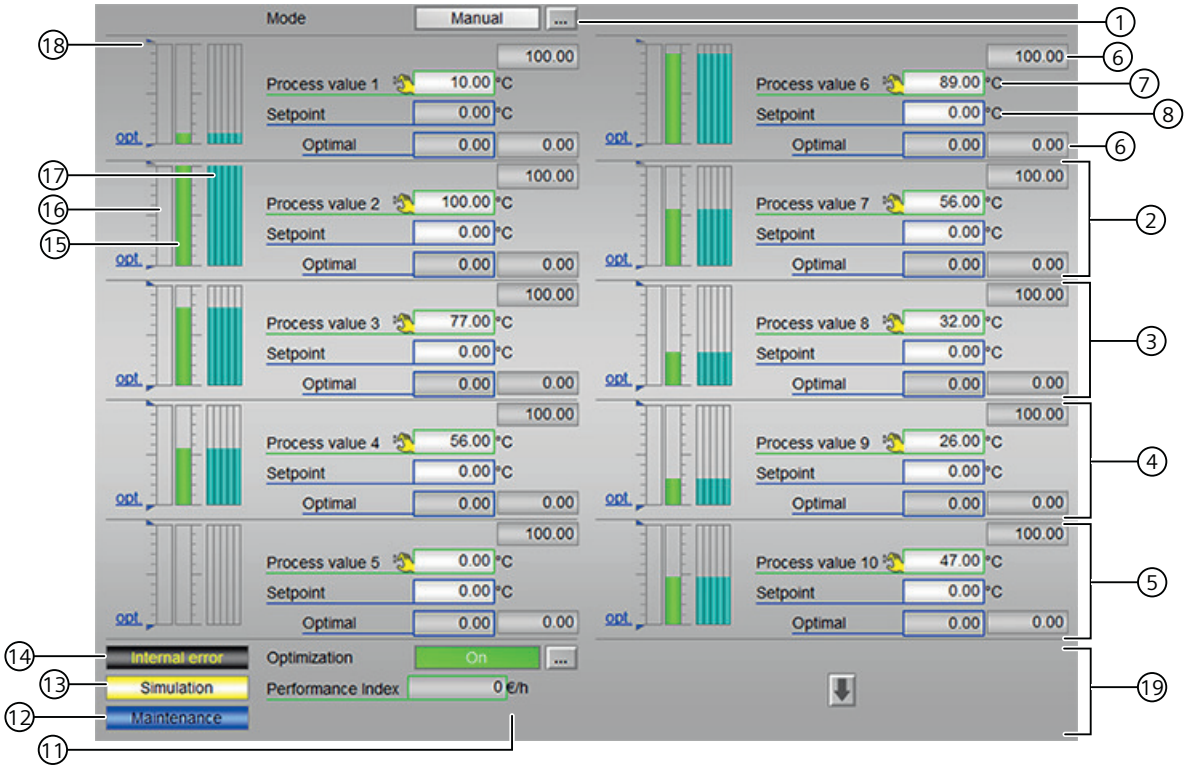
MPC10x10 trend view (Page 1273)

### 6.15.8.2 MPC10x10 standard view

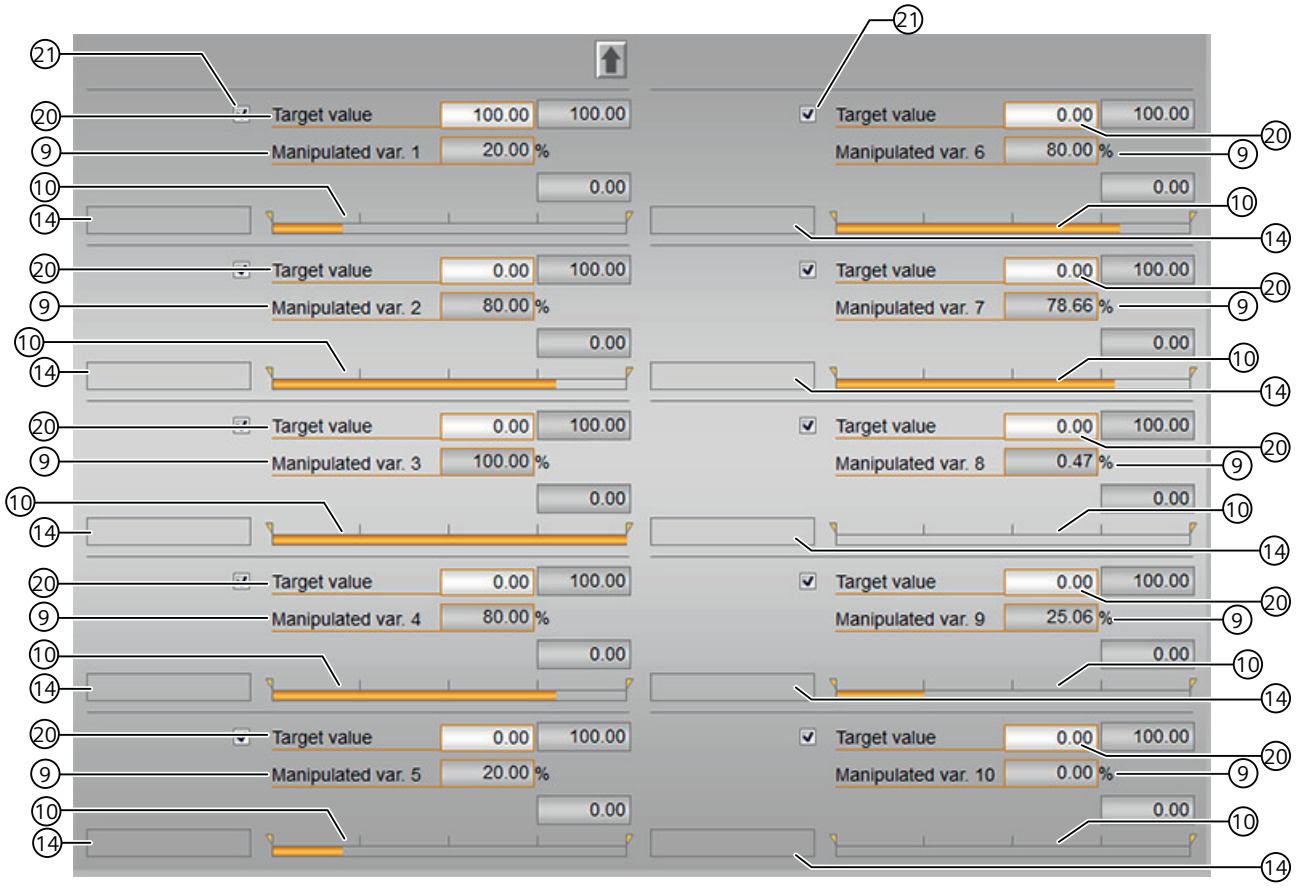
#### Standard view of MPC10x10

The standard view has an upper half and a lower half. You can change between the two halves with the arrow keys. The upper half shows all available controlled variable channels with their setpoints, while the lower half shows all available manipulated variable channels.

**Upper screen half (controlled variables)**




**Lower screen half (manipulated variables)**




## (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 79)
- Automatic mode (Page 79)
- Out of service (Page 77)

For information on switching the operating mode, refer to the section

Switching operating states and operating modes (Page 289).

## (2), (3), (4) and (5) Displaying and switching for values for channels 1 to 10

This area always has the same layout for channels 1 to 10:

**(6) High and low scale range for the process value**

These values provide information on the display range for the bar graph of the process value. The scale range is defined in the engineering system. For more information refer Bar graph (Page 376).

**(7) Displaying and changing the process value including signal status**

This area shows the current process value with the corresponding signal status.

The process value is normally displayed and cannot be operated. Process values in the faceplate can only be changed within the context of internal block simulation.

In the object properties (I/Os PVx_Out > Identifier) of the block in CFC, you can specify the text to be displayed for these parameters.

**(8) Displaying and changing the setpoint including signal status**

This area shows the current setpoint with the corresponding signal status. Refer to the Changing values (Page 291) section for information on changing the setpoint.

### (11) Navigation button for switching to the standard view of any faceplate

Use this navigation button to open the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234) .

### (12) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on the display area for states of the block is available in the Release for maintenance (Page 70) .

### (13) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"
- "Process excitation"
  The automatic process trigger is fed forward using the upstream block AutoExcitation for recording learning data for the MPC configurator. The manipulated variable step changes are added to the manipulated values 1 to 4 according to schedule. Avoid external disturbances to the process while the process trigger is running. The manipulated variables can be changed manually while the process trigger is running.

You can find additional information on this in the section Simulating signals (Page 64).

### (14) Display area for block states

- "Internal error"
- "Invalid Signal"

### (15) Display area for block states

- "Tracking on"

### (16) Bar graph for the process value 1

There is a bar graph for the process value for every channel 1 to 10.

This area shows the current process value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (17) Bar graph for the setpoint 1

There is a bar graph for the setpoint for every channel 1 to 10.

This area shows the current setpoint in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (18) Prediction of free movement

This area shows you the prediction of free movement in the form of a bar graph. For each channel from 1 to 10, there is a bar graph for the prediction of free movement, that is, for the future behavior of the process within the overall prediction horizon, under the assumption that all manipulated variables are frozen at their current values.

This is why the prediction of free movement is only displayed in manual mode or in automatic mode with active tracking function.

The value range of the bar graph matches the value range of the respective setpoint and current value bar.

Functions of MPC10x10 (Page 1243)

### (19) Displaying the limits

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the engineering system (ES).

### (20) Steady state operating point optimization

Activate the optimization using the button at bottom right. Activation means that the optimized setpoints SP1Out...SP4Out are actually used instead of the SP1...SP4 setpoints specified in the faceplate for closed-loop control. (The actual calculation of the optimum setpoints is not depending on this, and is only performed if one of the input variables for the optimization has changed.) The current value the economic performance index J appears in the display field below.

When optimization is enabled, the optimum setpoints are displayed on the setpoint bar as small, horizontal lines and highlighted with the abbreviation "opt.". The numerical values of the optimum setpoints are then displayed left of the input fields for the setpoints.

## Lower screen half (manipulated variables)

### (9) Displaying and changing the manipulated variable including signal status

This area shows the current manipulated variable with the corresponding signal status. Refer to the Changing values (Page 291) section for information on changing the manipulated variable. You can only make a change in manual mode.

In the object properties (I/Os MVx > Identifier) of the block in CFC, you can specify the text to be displayed for these parameters.

**(10) Bar graph for the manipulated variable with limit display**

This area shows the current manipulated variable in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES):

- Limits: `MVxHiLim` and `MVxLoLim`
- Display area: `MVxManHiLim` and `MVxManLoLim`

### (14) Display for states of the manipulated channel

There is a display for the state of the manipulated channel for channels 1 to 10: the status display only appears if the respective channel is tracked:

- "Tracking"

### (20) Displaying and changing the target value for the manipulated variable

This area shows the current target value for the manipulated variable. Refer to the section Changing values (Page 291) for more information on changing the target value. You can make a change only in the "Automatic" mode and also only if `OptimAct` = 0. This field is not displayed in "Manual" mode.

### (21) Activate/Deactivate MV Target

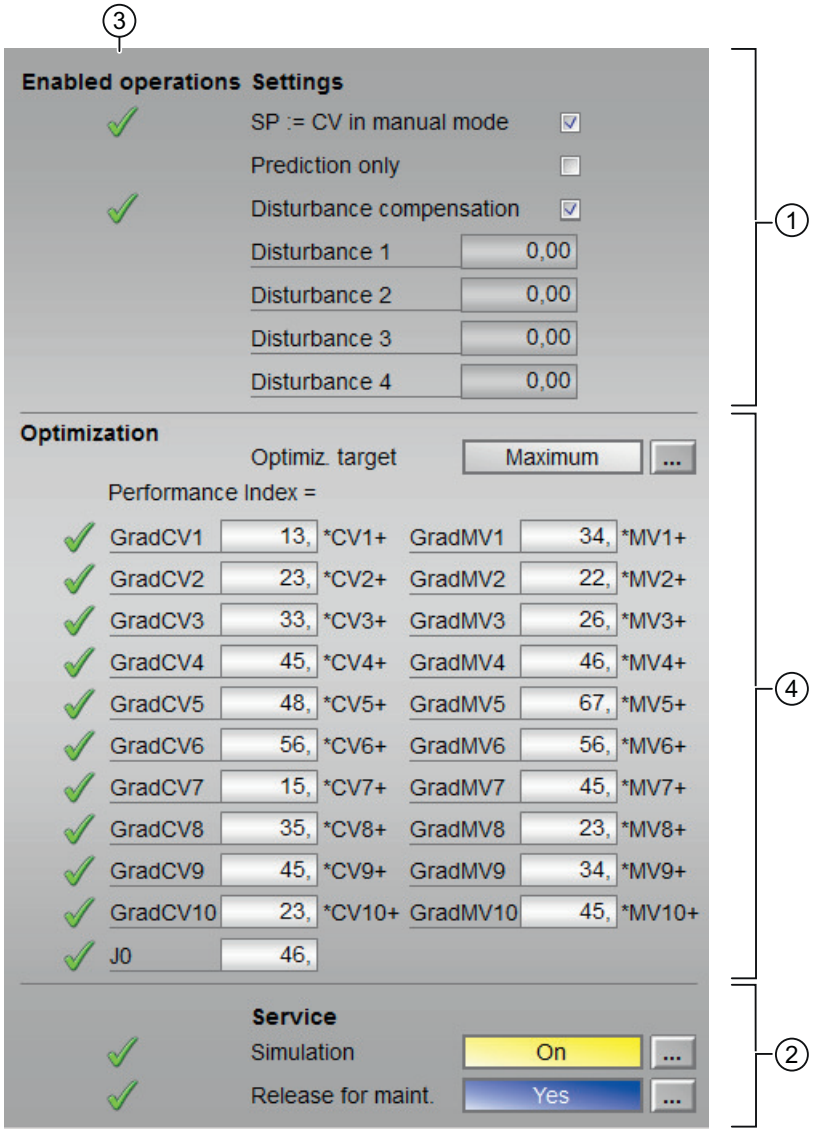
When activated, `MV target` value entered by operator will be considered else target value will be tracked to the `MV value`.

### See also

MPC10x10 preview (Page 1271)

### 6.15.8.3 MPC10x10 parameter view

#### Parameter view of MPC10x10




#### (1) Settings

You can activate the following functions for the controller in this area:

- "SP := CV in manual mode": ☑ Bumpless switchover from "manual mode" to "automatic mode"
- "Prediction only" activate this special "operating mode" by selecting the check box. The controller then only listens to the process and indicates what it would like to do in the next sampling step (i.e. which manipulated variables it would output in the next sampling step) without actively intervening in the process

- "Disturbance compensation": ☑ Select feedforward disturbance compensation
- "Disturbance" DV1...DV4 depending on the number of configured measurable disturbances

You cannot change the disturbance values, they can only be displayed.

In the object properties (I/Os DVx > Identifier) of the block in CFC, you can specify the text to be displayed for this parameter.

### (2) Service

You can select the following functions in this area:

- "Simulation"
- "Release for maintenance"

Refer to the Switching operating states and operating modes (Page 289) section for more on this.

You can find information on this area in the section:

- Simulating signals (Page 64)
- Release for maintenance (Page 70)

### (3) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

### (4) Optimization

**Direction of the optimization (minimize or maximize)**

By default, the optimizer seeks to maximize the performance function, in the assumption that it is dealing with economic yield. If you want to search a minimum, however, because you are dealing with costs or consumption values, click this button.

**Specification of performance index for the operating point optimization**

The performance index consists of a weighted sum of all manipulated and controlled variables. For each manipulated variable and controlled variable, enter the appropriate weighting factor, i.e. the coefficient of the gradient vector. Zero means that the value of the corresponding manipulated variable or controlled variable has no direct influence on the economic yield. If the controller has less than 10 manipulated variables or controlled variables, the irrelevant variables are hidden automatically.

J0 is the part of the performance index which does not depend on the controlled and manipulated variables of the MPC, e.g. fixed costs or costs calculated otherwise. Although this part of costs cannot be influenced during controller internal optimization, including it in the summation allows realistic numerical values to be displayed for the total costs.

### See also

MPC10x10 views (Page 1261)

MPC10x10 standard view (Page 1261)

MPC10x10 trend view (Page 1273)

MPC10x10 preview (Page 1271)

## 6.15.8.4 MPC10x10 CV parameter view

### CV parameter view for MPC10x10

The CV parameter view is a table which contains a row for each control channel:

| Channel setpoint | H range | H range optimiz. | Operator specific | Optimal setpoint | L range optimiz. | L range | Dead band | Unit | Prefilter [s] |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 100,00 | 0,00 | 110,00 | 100,00 | 0,00 | 0,00 | 0,00 | °C | 0,00 |
| 2 | 600,00 | 0,00 | 125,00 | 125,00 | 0,00 | 0,00 | 0,00 | °C | 0,00 |
| 3 | 300,00 | 0,00 | 95,00 | 95,00 | 0,00 | 0,00 | 0,00 | °C | 0,00 |
| 4 | 400,00 | 0,00 | 135,00 | 135,00 | 0,00 | 0,00 | 0,00 | °C | 0,00 |
| 5 | 100,00 | 0,00 | 140,00 | 100,00 | 0,00 | 0,00 | 0,00 | °C | 0,00 |
| 6 | 200,00 | 0,00 | 126,00 | 126,00 | 0,00 | 0,00 | 0,00 | °C | 0,00 |
| 7 | 100,00 | 0,00 | 7,85 | 7,85 | 0,00 | 0,00 | 0,00 | °C | 0,00 |
| 8 | 100,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | °C | 0,00 |
| 9 | 100,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | °C | 0,00 |
| 10 | 100,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | °C | 0,00 |

The channel number is at the far left. The physical unit applies to all values of this control channel except for the prefilter. The time constant of the prefilter is specified for all channels in seconds.

### (1) Displaying and changing the limit parameters for the setpoint

You can change the following setpoint parameters for the relevant control channel in each row:

- "H range": High limit for setpoint operation
- "H range optimization": High limit for optimizing the setpoint
- "Operator specification": Display of the setpoint specified in the standard view, cannot be operated here.

- "Dead band": Error signal generation and dead band (Page 217), Error signal generation and dead band section
- "Optimal setpoint": Calculated by the optimization, cannot be operated
- "L range optimization": Low limit for optimizing the setpoint
- "L range": Low limit for setpoint operation
- "Prefilter": TimeTrig functions (Page 2184), Setpoint filter section

The numerical values for limits are arranged from left to right in descending order. The three values per channel relevant for the optimization are highlighted in blue.

You can find additional information on this in the section Changing values (Page 291) .

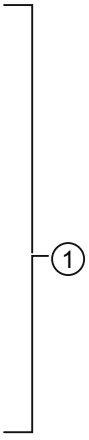
The enabled operations for the setpoint parameters are displayed in the preview.

### 6.15.8.5 MPC10x10 MV parameter view

#### MV parameter view for MPC10x10

The MV parameter view is a table which contains a row for each manipulated variable channel:

| MV channel | H range | L range | Unit | Gradient [Unit/s] |
|---|---|---|---|---|
| 1 | 100,00 | 0,00 | % | 100,00 |
| 2 | 100,00 | 0,00 | % | 100,00 |
| 3 | 100,00 | 0,00 | % | 100,00 |
| 4 | 100,00 | 0,00 | % | 100,00 |
| 5 | 100,00 | 0,00 | % | 100,00 |
| 6 | 100,00 | 0,00 | % | 100,00 |
| 7 | 100,00 | 0,00 | % | 100,00 |
| 8 | 100,00 | 0,00 | % | 100,00 |
| 9 | 100,00 | 0,00 | % | 100,00 |
| 10 | 100,00 | 0,00 | % | 100,00 |




The channel number is at the far left. The physical unit applies to the high and low limit of the respective channel. The gradient is limited in the respective manipulated variable unit per second.

#### (1) Displaying and changing the limit parameters for the manipulated variable

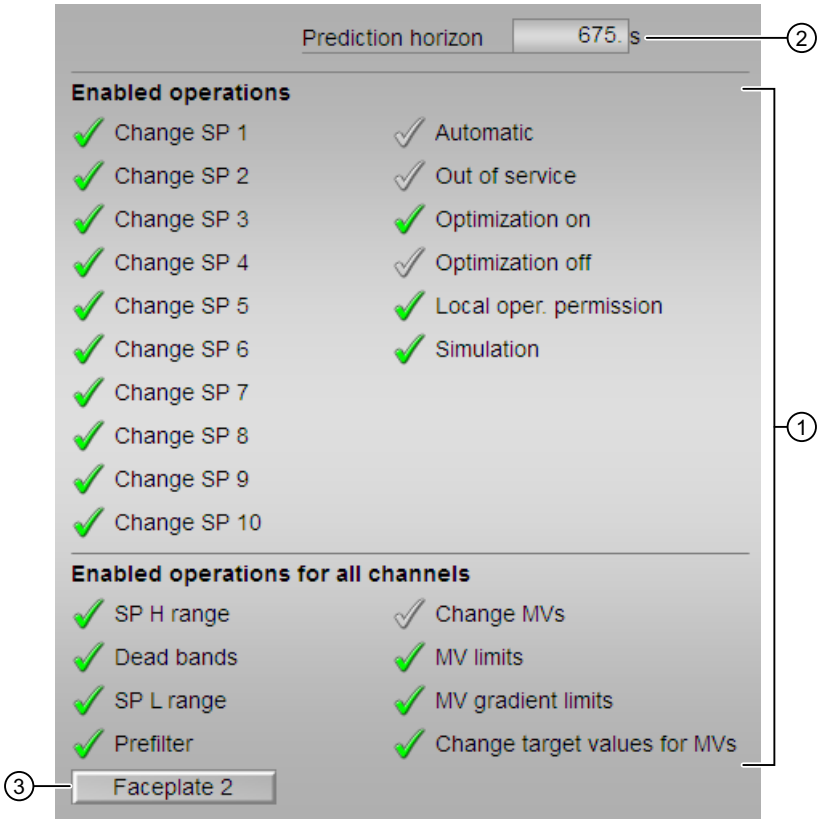
You can change the following parameters for the manipulated variable in this area:

- "H range": Upper limit of the manipulated variable for automatic mode
- "L range": Low limit of manipulated variable for automatic mode
- "Gradient limit": Maximum (absolute) change in the manipulated variable per second

The enabled operations for the setpoint parameters are displayed in the preview.

### 6.15.8.6 MPC10x10 preview

#### Preview for MPC10x10




#### (1) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block. The lower screen half shows operator permissions which apply to all manipulated and control channels.

The upper screen half shows operator permissions which are not channel-based on the left, and channel-specific operator permissions for the individual setpoints on the right.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

The following enabled operations for parameters are shown here:

- "Automatic": You can switch to "automatic mode".
- "Out of service": You can switch to "Out of service" operating mode.
- "Optimization on": You can switch on the optimization.
- "Optimization off": You can switch off the optimization.
- "Local operator permission": Use the ← button to switch to the standard view of the OpStations block. You can find additional information on this in the section Operator permissions (Page 285) .
- "Change SP1..10": You can change the setpoint 1..10
- "Change MVs": You can change the manipulated variables

**Note**

The OS operator must always be able to switch to "manual mode". For this reason, there is no special operator permission for switching to "manual mode" in the faceplate.

The following operator permissions apply to all manipulated channels.

- "Change MVs": You can change the manipulated variables in manual mode
- "MV limits": You can change the high and low limits for all manipulated variables.
- "MV gradient limits": You can change the gradient limits for all manipulated variables.
  The following operator permissions apply to all control channels.
  - "SP high limits": You can change the high limits for all setpoints.
  - "Dead bands": You can change the dead bands for all controlled variables.
  - "SP low limits": You can change the low limits for all setpoints.
  - "Prefilter": You can change the time constants of the prefilter for all setpoints.
- "Change target values for MVs": You can change the target values for all manipulated variables.

### (2) Prediction horizon

The prediction horizon specifies how far the controller looks into the future in its calculations. The value is set in the MPC Configurator and displayed in the faceplate for informational purposes.

### (3) Navigation button for switching to the standard view of any faceplate

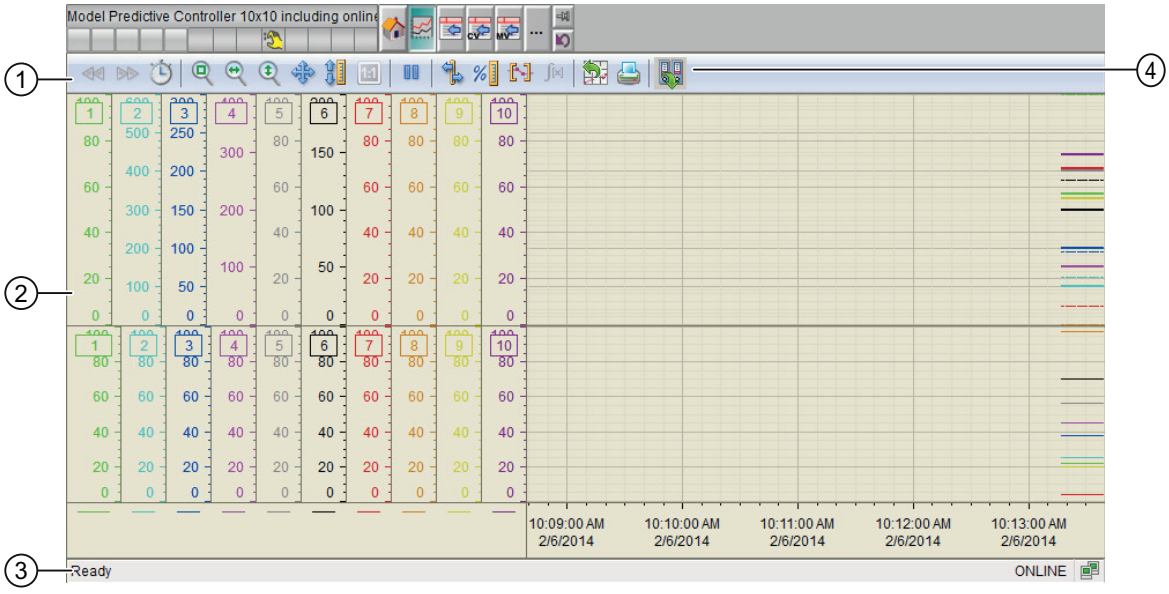
Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234) .

### 6.15.8.7 MPC10x10 trend view

## MPC10x10 trend view

There is a block-specific trend view for MPC10x10 which is supplied as file @PG_APL_TrendMPC_L.pdl and which you can modify if necessary.




**(1)** Toolbar

**(2)** Display area for trends

**(3)** Status bar

**(4)** Button for switching between archive tags and online tags. The status bar shows if the trend view is working with online data or archive data.

The Export button is only visible and operable with the "Higher-level process control" operating permission.

For additional information about the trend view, refer to the *WinCC Information System* Online Help.

The trend view is divided into two screen halves.

The upper screen half shows all controlled variables with their associated setpoints. The setpoint is shown in the same color as the associated process value to allow the assignment to be identified straight away. Setpoints are dashed lines, process values are bold lines. If a controlled variable is exactly on the setpoint, it hides the setpoint.

The lower screen half shows all manipulated variables.

Both screen halves use the same color sequence for the individual channels. The sequence starts at channel 1 with green (standard color for the process value with the PID controller) and then goes through the color spectrum from top to bottom as far as gray and black. From channel 7 (red), the rest of the spectrum is run through from top to bottom. Each channel has its own y-axis in the corresponding color.

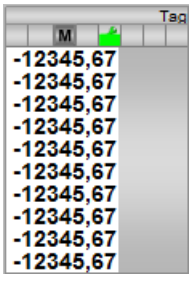
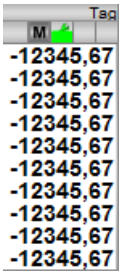
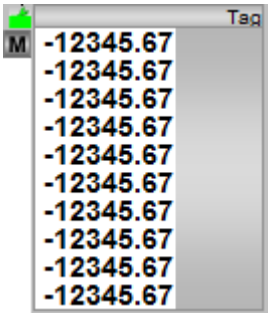
### 6.15.8.8 Block icon for MPC10x10

## Block icons for MPC10x10

A variety of block icons are available with the following functions:

- Process tag type
- Operating modes
- Signal status, release for maintenance
- Tracking
- Memo display
- Process value (black, with and without decimal places)

The display of the tag name can be activated centrally for all block instances with the WinCC variable @APLShowTag:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | Narrow, without units |
|  | 3 | Block icon in the full display |

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 267)
- Operation via the block icon (Page 269)
- Block icon structure (Page 260).

# 6.16 KalFilt - State observer

## 6.16.1 Description of KalFilt

### Object name (type + number) and family

Type + number: FB 1925

Family: Control

### Area of application for KalFilt

The Kalman filter is a non-linear stochastic state observer. It forms a framework for parallel simulation of a dynamic process model, whose states are compared with real data in each cycle. During runtime, all internal state variables may be read from the model and used as a substitute for real measured values.

The state observer thus allows to determine the physical values of a process unit, which are not accessible or only measurable with high effort from other easily available measurements, for example, temperatures and mass flows.

It can therefore be used as a soft sensor, that is a piece of software that estimates values which are not measured.

In addition to its use as a state observer, it is also used as a model-based filter. The measured variables calculated from the estimated states are not subject to measurement noise and therefore generally run smoother than noisy measurements.

### How it works

#### The extended Kalman filter

The block operates as a state observer, in which the algorithm of the extended Kalman filter is implemented, and the online calculation of the internal states of non-linear dynamic systems is performed. Both, the mathematical structure of the underlying dynamic system and the properties of the stochastic disturbances must be known.

The term "filter" indicates that the algorithm is able to reduce measurement noise and other disturbances of stochastic data.

The Kalman filter is an approach to solve the general problem of estimating the states of a time-discrete process, which is described by linear stochastic differential equations and is a generalization of the classical state observer (Luenberger observer) in this respect. The estimation problem is extended to non-linear processes that result from a non-linear state space model:

$$\underline{x}_{k+1} = \underline{f}\left(\underline{x}_k, \underline{u}_k\right) + \underline{w}_k$$

$$\underline{y}_k = \underline{h}\left(\underline{x}_k\right) + \underline{v}_k$$

| Symbols | Meaning |
|---|---|
| $\underline{x}_k$ | Vector of the (inner) state variables at the time $t_k$ |
| $\underline{u}_k$ | Vector of the input variables at the time $t_k$ |
| $\underline{y}_k$ | Vector of the measurable output variables at the time $t_k$ |
| $\underline{\Phi}_k$ | Dynamic matrix (Jakobi matrix), refer to FB KalMod |
| $\underline{H}_k$ | Output matrix (Jakobi matrix), refer to FB KalMod |
| $\underline{P}_k$ | Estimation error covariance at the time $t_k$ |
| $\underline{K}_k$ | Kalman gain at the time $t_k$ |
| $\underline{Q}_k$ | Covariance of the process noise at the time $t_k$ |
| $\underline{R}_k$ | Covariance of the measurement noise at the time $t_k$ |
| $\underline{w}_k$ , $\underline{v}_k$ | Stochastic disturbances |

For the Kalman algorithm, *a priori* knowledge about the process to be observed must be provided to the block. This includes the covariance of the process noise $\underline{Q}_k$, the covariance of the observation noise $\underline{R}_k$, and the initial values (initialization) for the *a priori* state estimate $\underline{\hat{x}}_0^-$. The *a priori* estimation error covariance $\underline{P}_0^-$ is initialized with 0.

For more details on the configuration of the filter, refer to the *Kalman Configurator User Documentation*. The configuration, identification, simulation, and validation of the filter are performed in the Kalman Configurator.

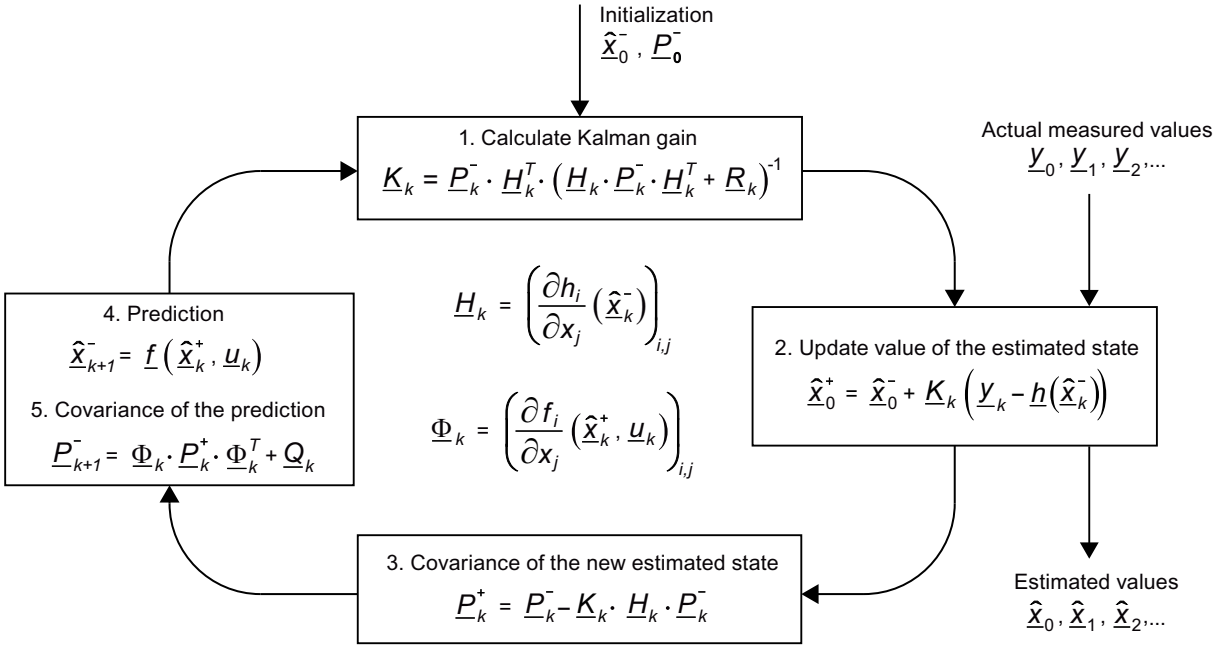



Figure 6-1 Implemented sequence of the extended Kalman filter

The sequence of the extended Kalman filter (EKF) corresponds to the linear filter: via the Kalman Gain $\underline{K}_k$ the predicted (*a priori*) state estimate $\underline{\hat{x}}_k^-$ is corrected using the actual measured values $\underline{y}_k$, which results in the corrected (*a posteriori*) state estimate $\underline{\hat{x}}_k^+$. Only two additional intermediate steps for calculating the Jacobi matrices $\underline{H}_k$ and $\underline{\Phi}_k$ are necessary for local linearization of non-linear models. The Jacobi matrices are calculated in the FB KalMod.

**Consideration of sporadic laboratory measurements**

If sporadic laboratory samples of relevant state variables are extracted from a process, the integration of these laboratory measurements in the extended Kalman filter can lead to significant improvements in estimation accuracy, since the unknown state variables can be compared with the laboratory results. The sample is taken sporadically. Once laboratory results are received, the laboratory values are provided to the EKF. The algorithm takes into account that the laboratory result is associated with a previous process state, which belongs to the past time instant where the sample was taken. For the usage of laboratory measurements, particular engineering steps are necessary. These steps are described in the section "Additional configuration tasks for inclusion of the laboratory measurements" under "Configuration".

## Operating principle

The corrected states $\underline{\hat{x}}_k^+$ (`X1`...`X15`) are used for application as a state observer. These can be used for visualization, interconnections to further calculations, or feedback for control loops.

The output variables $\underline{\hat{y}}_k^+ = \underline{h}\left(\underline{\hat{x}}_k^+\right)$ (`Y1Filt`...`Y7Filt`) calculated from the corrected states are used for application as a model-based filter.

The advantage compared to a simple low-pass filter is that the influences of the inputs are provided without additional delay to the output variables via the model. The signal status of the filtered output values `YiFilt` are based on the worst signal status of the states.

To assess the quality of the Kalman filter, the following deviations can be used:

- The deviations of the predicted output values from the measured variables:
  $\underline{e}_k^- = \underline{y}_k - \underline{h}\left(\underline{\hat{x}}_k^-\right)$ (`EY1`...`EY7`)
- The deviations of the predicted state variables from the laboratory measurements:
  $\underline{e}_{nk}^- = \underline{y}_{nk} - \underline{h}_n\left(\underline{\hat{x}}_k^-\right)$ (`EYn1`...`EYn5`)

The deviations are usually caused by modeling inaccuracies and unknown disturbances (i.e. disturbances not considered in the model).

In many applications, there are not measurable outputs that can be calculated using algebraic equations from state and input variables, and which are relevant for the practical use of the Kalman filter, for example, heat flow generated by exothermic chemical reactions or specific conversion rates in organic fermenters. These variables can be defined as non-measurable process output variables (`Z1Out`...`Z5Out`) in the Kalman configurator, for which there is a respective output equation, but no measured values.

For application of the filter block as a process simulator, the input variables `Ui` are interconnected to the actual input values to simulate the behavior of the process and get simulated measured values at the output variables `YjFilt`. Since there are no measurements for the process output variables `Yj`, the corresponding inputs at the block are connected to the output variables. The elements of the Q matrix are all set to zero, that is no model uncertainty is assumed for the simulation.

### Kalman Configurator

Using the Kalman Configurator, you can conveniently configure the process model and specify all model-specific settings for the Kalman filter. As a result, the tool generates the source code of the FB KalMod. Once you have compiled the block, you can download it to the automation system. For more details, refer to the *Kalman Configurator User Documentation*.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

Perform the following configuration tasks:

- Interconnect the input and output variables `U1`...`U7` and `Y1`...`Y7`
- Interconnect `UiLimScale`, `UiUnit` (i = 1...`NumberU`) and `YiLimScale`, `YiUnit` (i = 1...`NumberY`) and specify input limits (`UiLimScale.High/.Low`) and the measurement limits (`YiLimScale.High/.Low`)
- Specify self-explaining variable names for all configured inputs, measurable and non-measurable outputs, and state variables at the text attributes (identifiers)
- Select the associated process model in FB KalMod through `ModelNo` (1...5)
- Interconnect the initial values for the configured state variables. This connection is only necessary if the initial values are not constant, but time-varying, and a signal source for the actual values is available.
- Record measured data and configure the Kalman filter through the Kalman Configurator (refer to the *Kalman Configurator User Documentation*)

To compile the SCL sources generated by the configurator, the following user-defined data types are needed, which have to be copied from APL to the function block folder of the user project:

- UDT 43: Vector15
- UDT 44: Matrix15
- UDT 45: Kal_DB
- UDT 46: StcVector15
- UDT 47: StcMatrix15
- UDT 48: StcKalDB

The data types which you have declared must not be within this number range.

- In the "Component view" of SIMATIC Manager, right-click on the folder "Sources", select "Insert new object" and then select "External source".
- Double-click the SCL source to open/edit the code and click "Compile and load" to upload it to the CPU.
- Set the input variable `LoadParam` = 1 at the KalFilt instance in the CFC to load the model in the filter.

Additional configuration tasks for inclusion of the laboratory measurements:

- Enable bit for lab measurements (`SampleEn` = 1)
- The number of lab data (`NumberYn`) is already determined in the configurator.

Laboratory data can be inserted using the faceplate or can be provided by interconnection. Therefore, the respective signal selectors `***_LiOp` have to be set, whereby `***_LiOp` = 1 means that the link is relevant, and `***_LiOp` = 0 means that the operation in the faceplate is relevant. Both the cases are described separately as follows:

- Inserting laboratory data in the faceplate:
  - Select the signal selectors (`SaPo_LiOp` = 0, `SaRe_LiOp` = 0)
  - The operation of the faceplate to insert laboratory data is described in section KalFilt standard view (Page 1293).
- Providing laboratory data by interconnection:
  - Select the signal selectors (`SaPo_LiOp` = 1, `SaRe_LiOp` = 1)
  - At the time of taking the lab sample, set the input variable `SaPo_Ext` = 1. This command is detected via rising edge of `SaPo_Ext` and is acknowledged by setting the binary output variable `SampExtracted` = 1.
  - After laboratory analysis, provide the results to the input variables `Yn1_Ext`...`Yn5_Ext` and set the input variable `SaRe_Ext` = 1. This command is detected via rising edge of `SaRe_Ext` and is acknowledged by setting the binary output variable `LabResIncluded` = 1.
  - `Feature.Bit2` defines if the binary input variables are reset by the function block itself. If they should not be reset automatically, they have to be reset by interconnection, after the respective order was acknowledged by setting the respective binary output variable.

Diagonal elements of the covariance matrix of the measurement noise `Rn11`...`Rn55` are defined in the model FB from the Kalman Configurator.

Determining the monitoring time `TimeMon`:

- When this time has expired, waiting for the laboratory test results is terminated. It takes at least one cycle of the process model between the period when the lab sample is taken and the laboratory results are received.

The library contains the following blocks:

| Icon | Address | Data type | Comment |
|---|---|---|---|
| KalFilt | FB 1925 | FB 1925 | State observer based on the algorithm of the extended Kalman filter. |
| KalFunct | FB 1926 | FB 1926 | Multiplexer for calling the correct model |
| KalMod1...KalMod5 | FB 471...FB 475 | FB 471...FB 475 | Process model including Jacobi matrices and filter parameters, created in the configurator |
| MxAdd | FC 466 | FC 466 | Adds two matrices (used internally) |
| MxCBnd | FC 467 | FC 467 | Merging by column (columnbind) of two matrices (used internally) |
| MxInv | FC 468 | FC 468 | Returns an inverse of a matrix (used internally) |
| MxMul | FC 469 | FC 469 | Multiplies two matrices (used internally) |
| MxRBnd | FC 470 | FC 470 | Merging by row (rowbind) of two matrices (used internally) |
| MxSub | FC 471 | FC 471 | Subtracts two matrices (used internally) |
| MxTrans | FC 472 | FC 472 | Returns a transpose of a matrix (used internally) |
| MxUnity | FC 473 | FC 473 | Generates an identity matrix (used internally) |

| Icon | Address | Data type | Comment |
|---|---|---|---|
| MxZero | FC 474 | FC 474 | Generates a zero matrix (used internally) |
| Vector15 | UDT 43 | UDT 43 | Data type for vector calculations |
| Matrix15 | UDT 44 | UDT 44 | Data type for matrix calculations |
| Kal_DB | UDT 45 | UDT 45 | Data type for user data |
| StcVector15 | UDT 46 | UDT 46 | Structure to deliver arrays of type Vektor15 |
| StcMatrix15 | UDT 47 | UDT 47 | Structure to deliver arrays of type Matrix15 |
| StcKal_DB | UDT 48 | UDT 48 | Structure to deliver arrays of type Kal_DB |

## Startup characteristics

At the start of a run or a batch, the algorithm must be initialized and all states must be set to their initial values. This initialization is triggered by the binary input variable `Reset`, which is automatically reset after executing the initialization and is also accessible in the faceplate.

If the initial values are read from the model FB generated by the Kalman Configurator during initialization, the binary input variable `LoadParam` should be used instead, which is also automatically reset after a successful read operation, but is not accessible in the faceplate. The current values set at the corresponding input variables or in the faceplate are overwritten (for example, initial values of the states) when the parameters are loaded from the FB.

When the CPU starts up, the block is reinitialized and the parameters are loaded from the FB (in accordance with `LoadParam`).

## Status word allocation for `Status1` parameter

You can find description of each parameter in the section KalFilt I/Os (Page 1287).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct` |
| 4 | `OosLi` |
| 5 | Not used |
| 6 | `OnAct` |
| 7 | `SampleEn`, enable for introducing the laboratory values |
| 8 | `SamplePoint`, laboratory sample taken from process |
| 9 | `SampleResult`, lab results incorporated |
| 10 | Laboratory results are bad → lab results are ignored |
| 11 | Waiting period for laboratory results has expired |
| 12 | Laboratory results adapted to limits |
| 13 | `SaPo_ExtAct`, external lab sampling |
| 14 | `SaRe_ExtAct`, external lab measurement |
| 15 | Not used |
| 16 - 25 | `UserAna1` to `UserAna10` interconnected |

| Status bit | Parameter |
|---|---|
| 26 - 30 | Not used |
| 31 | Internal error |

### See also

KalFilt modes (Page 1282)

KalFilt functions (Page 1284)

KalFilt error handling (Page 1286)

KalFilt messaging (Page 1292)

KalFilt block diagram (Page 1292)

## 6.16.2 KalFilt modes

### Operating modes of KalFilt

The block can be operated using the following modes:

- On (Page 77)
- Out of service (Page 77)

### "On"

You can find general information about the "On" mode in the section On (Page 77).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit5` =1 and `Feature.Bit5` =0 is as shown below:

| Output Parameters | **`Feature.Bit5 =0`** | **`Feature.Bit5 =1`** |
|---|---|---|
| X1.ST | 16#80 | 16#60 |
| X2.ST | 16#80 | 16#60 |
| X3.ST | 16#80 | 16#60 |
| X4.ST | 16#80 | 16#60 |
| X5.ST | 16#80 | 16#60 |
| X6.ST | 16#80 | 16#60 |
| X7.ST | 16#80 | 16#60 |
| X8.ST | 16#80 | 16#60 |
| X9.ST | 16#80 | 16#60 |
| X10.ST | 16#80 | 16#60 |

| Output Parameters | Feature.Bit5 =0 | Feature.Bit5 =1 |
|---|---|---|
| X11.ST | 16#80 | 16#60 |
| X12.ST | 16#80 | 16#60 |
| X13.ST | 16#80 | 16#60 |
| X14.ST | 16#80 | 16#60 |
| X15.ST | 16#80 | 16#60 |
| U1Out.ST | 16#80 | 16#60 |
| U2Out.ST | 16#80 | 16#60 |
| U3Out.ST | 16#80 | 16#60 |
| U4Out.ST | 16#80 | 16#60 |
| U5Out.ST | 16#80 | 16#60 |
| U6Out.ST | 16#80 | 16#60 |
| U7Out.ST | 16#80 | 16#60 |
| Y1Out.ST | 16#80 | 16#60 |
| Y2Out.ST | 16#80 | 16#60 |
| Y3Out.ST | 16#80 | 16#60 |
| Y4Out.ST | 16#80 | 16#60 |
| Y5Out.ST | 16#80 | 16#60 |
| Y6Out.ST | 16#80 | 16#60 |
| Y7Out.ST | 16#80 | 16#60 |
| Yn1Out.ST | 16#80 | 16#60 |
| Yn2Out.ST | 16#80 | 16#60 |
| Yn3Out.ST | 16#80 | 16#60 |
| Yn4Out.ST | 16#80 | 16#60 |
| Yn5Out.ST | 16#80 | 16#60 |
| Y1Filt.ST | 16#80 | 16#60 |
| Y2Filt.ST | 16#80 | 16#60 |
| Y3Filt.ST | 16#80 | 16#60 |
| Y4Filt.ST | 16#80 | 16#60 |
| Y5Filt.ST | 16#80 | 16#60 |
| Y6Filt.ST | 16#80 | 16#60 |
| Y7Filt.ST | 16#80 | 16#60 |
| Z1Out.ST | 16#80 | 16#60 |
| Z2Out.ST | 16#80 | 16#60 |
| Z3Out.ST | 16#80 | 16#60 |
| Z4Out.ST | 16#80 | 16#60 |
| Z5Out.ST | 16#80 | 16#60 |
| SaPo_ExtAct.ST | 16#80 | 16#60 |
| SaRe_ExtAct.ST | 16#80 | 16#60 |

## See also

Description of KalFilt (Page 1276)

KalFilt functions (Page 1284)

KalFilt error handling (Page 1286)
KalFilt I/Os (Page 1287)
KalFilt messaging (Page 1292)
KalFilt block diagram (Page 1292)

## 6.16.3 KalFilt functions

### Functions of KalFilt

The functions for this block are listed below.

### Forming the signal status for blocks

The block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

- Signal status for the estimated states values `X1`...`X15`:
  - When limited (to the high or low limit), the signal status 16#78 (uncertain, process-related) is set at the relevant state estimate.
  - If the REAL value of the input or output variables is "infinity" (+#INF and -#INF) or "not a number" (+#NAN and -#NAN), the signal status 16#28 (Bad) is assigned to all state variables.
  - If a signal status of the input variables (`U`) is bad, all state variables obtain the signal status `ST_Worst`.
  - If a signal status of the output variables (`Y`) is bad, the associated covariance element of the measurement noise `R` is set to 1.0e+10 and all state variables obtain the signal status 16#68 (uncertain, device related).
  - If a signal status of the lab measured values (`Yn`) is bad or the lab value cannot be shown in the REAL number field, all lab values are ignored. The signal status of the lab measured values has no affect on the signal status `ST_Worst`.
  - In all other cases, the state estimated values have the signal status 16#80 (Good).
- The worst signal status `ST_Worst` for the block is formed from the parameters (`U1.ST` to `U7.ST`).

### Configurable reactions using the `Feature` parameter

An overview of all the reactions that are provided by the `Feature` parameter is available in the section Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | AUTOHOTSPOT |

| Bit | Function |
|---|---|
| 2 | Resetting the commands for changing the mode (Page 172) |
| 5 | Changes signal status of outputs in OOS (Page 208) |
| 24 | Enabling local operator authorization (Page 169) |

## Operator permissions

The block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch the block to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch the block to "Out of service" mode |
| 4 | Not used |
| 5 | 1 = Operator can perform a block restart |
| 6 | 1 = Operator can change `SimUi` simulation values |
| 7 | 1 = Operator can change `SimYi` simulation values |
| 8 | 1 = Operator can enter the initial values `XiStart` for state variables `X1`...`X15` |
| 9 | 1 = Operator can change the high limit `XiHiLim` for state variables `X1`...`X15` |
| 10 | 1 = Operator can change the low limits `XiLoLim` for state variables `X1`...`X15` |
| 11 | 1 = Operator can enable the simulation function |
| 12 | Not used |
| 13 | 1 = Operator can enter lab values and operate `SaPo` and `SaRe` |
| 14 | 1 = Operator can specify the maximum duration for lab data and limits for errors in lab values |
| 15 | 1 = Operator can change the scaling of the bar graph display for the estimation error in the standard view of the faceplate |
| 16 - 31 | Not used |

## See also

Description of KalFilt (Page 1276)

KalFilt modes (Page 1282)

KalFilt error handling (Page 1286)

KalFilt I/Os (Page 1287)

KalFilt messaging (Page 1292)

KalFilt block diagram (Page 1292)

## 6.16.4 KalFilt error handling

### Error handling of KalFilt

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers in `ErrorNum`
- Output bit `ErrorPara` = 1
  The occurrence of an error is depicted at the faceplate using the symbol  as function block status.

### Overview of error numbers

The `ErrorNum` output parameter can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| –1 | Predefined value when the block is inserted; the block is not executed. |
| 0 | There is no error. |
| 2 | `SampleTime` < 0.001 or required sampling time for Kalman filter is less than the cycle time of the cyclic interrupt OB in which the block is integrated. |
| 3 | FB KalFunct called internally returns an error, for example, if `ModelNo` > 5 |
| 4 | Parameter assignment error `NumberU`/`NumberY`/`NumberX`/`NumberYn`/`NumberZ`, for example, number larger than possible |
| 6 | Error loading model FB from the Kalman Configurator. The following sources of error are possible:<br>• FB is not available<br>• FB is not Kalman Configurator FB<br>• Required matrix dimensions are not supported |
| 7 | Only during startup:<br>Initial values of the state variables are out of range |
| 21 - 27 | Signal status of the input variable `U1`...`U7` is bad |
| 30 | The value of an input variable `Y1`...`Y7` or `U1`...`U7` cannot be displayed in the REAL number field |
| 31 | Matrix inversion inside the Kalman filter algorithm fails. In this case, the internal filter states will not be updated according to current measurement data, but remain at their *a priori* values. If at the next sample step, with new measurement values, the matrix-inversion works correctly again, the `ErrorNum` is set back to zero. |

### See also

Description of KalFilt (Page 1276)

KalFilt modes (Page 1282)

KalFilt functions (Page 1284)

KalFilt I/Os (Page 1287)

KalFilt messaging (Page 1292)

KalFilt block diagram (Page 1292)

## 6.16.5 KalFilt I/Os

I/Os of KalFilt

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameters | Description | Type | Default |
|---|---|---|---|
| BatchEn | 1 = Enable allocation for Batch control | BOOL | 0 |
| BatchID | Current batch ID | DWORD | 16#00000000 |
| BatchName | Current batch designation | STRING[32] | '' |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| EYn1Lim...EYn5Lim | Limits for the difference between lab results and associated state values | REAL | 0.0 |
| EYPScale | Scaling of the bar graph display of the estimation error in the faceplate | REAL | 100.0 |
| Feature | I/O for additional functions (Page 1284) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| LoadParam | 1 = Restart KalFilt with loading of the parameters (including the initial values for the states) from the FB | BOOL | Normal oper. |
| ModelNo | Process model number (assignment of the appropriate process model in KalFunc) | INT | 1 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OnOp | 1=On Mode: On Mode by Operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameters | Description | Type | Default |
|---|---|---|---|
| OosOp | 1 = "Out of service", via OS operator | BOOL | 0 |
| OS_Perm | I/O for operator permissions (Page 1284) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| Q11...Q1515 | Diagonal elements of the covariance matrix *Q*, related to uncertainty of the differential equation for respective states X1... X15 | REAL | 0.0 |
| Reset | 1 = Reset the states X1...X15 to their initial values defined at the input parameters | BOOL | Normal oper. |
| R11...R77 | Diagonal elements of covariance matrix *R*, related to measurement noise of output variables Y1...Y7 | REAL | 0.0 |
| Rn11...Rn55 | Diagonal elements of covariance matrix *Rn*, related to measurement noise of lab results Yn1...Yn5 | REAL | 0.0 |
| SampleEn | Enable for sporadic laboratory measurements | BOOL | 0 |
| SampleTime | Sample time in seconds (assigned automatically) | REAL | 0.1 |
| SaPo_LiOp | Select lab sampling (internal/external):<br>1 = Via interconnection (external)<br>0 = Via operator (internal) | BOOL | 0 |
| SaPo_Int | 1 = Lab sampling internal (via operator) | BOOL | 0 |
| SaPo_Ext | 1 = Lab sampling external (via interconnection) | BOOL | 0 |
| SaRe_LiOp | Select lab result (internal/external):<br>1 = Via interconnection (external)<br>0 = Via operator (internal) | BOOL | 0 |
| SaRe_Int | 1 = Lab result internal (input by operator) | BOOL | 0 |
| SaRe_Ext | 1 = Lab result external (values comes via interconnection) | BOOL | 0 |
| SelFp1 | 1 = Call a block saved in this parameter as an additional faceplate in the standard view | ANY | |
| SelFp2 | 1 = Call a block saved in this parameter as an additional faceplate in the preview | ANY | |
| SimOn | Switch on/off of the simulation | BOOL | 0 |
| SimU1...SimU7 | Simulation values for U1...U7 | REAL | 0.0 |
| SimY1<br>...<br>SimY7 | Simulation values for Y1...Y7 | REAL | 0.0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| TimeMon | Monitoring time for sampling in minutes | INT | 60 |
| U1...U7 | Input variables | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| U1Unit...U7Unit | Unit for input variables U1...U7 | INT | 1342 |

| Parameters | Description | Type | Default |
|---|---|---|---|
| U1LimScale...U7LimScale | Limit of the value range and scaling for faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| UserAna1...UserAna10 | Arbitrary analog values for display in the block icon | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UA1Unit...UA10Unit | Unit for UserAna1...UserAna10 | INT | 1001 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |
| X1LimScale...X15LimScale | High and low limit for X1...X15 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| X1Start...X15Start | Initial value for the state variables X1...X15 | REAL | 0.0 |
| X1Unit...X15Unit | Unit for the state variables X1...X15 | INT | 1001 (°K) |
| Y1...Y7 | Measurable output variables | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Y1LimScale...Y7LimScale | Limitation of the value range of Y1...Y7 and scaling for faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| Y1Unit...Y7Unit | Unit for the output variables Y1...Y7 | INT | 1001 |
| Yn1_Ext...Yn5_Ext | External lab result (input for interconnection) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Yn1_Int...Yn5_Int | Internal lab result Yn1_Int...Yn5_Int (input by operator) | REAL | 0.0 |
| Yn1Unit...Yn5Unit | Unit for the lab result Yn1...Yn5 | INT | 1342 (%) |
| Yn1LimScale...Yn5LimScale | Limitation of the value range of Yn1...Yn5 and scaling for faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| Z1Unit...Z5Unit | Unit of measurement for Z1...Z5 | INT | 1001 |

## Output parameters

| Parameters | Description | Type | Default |
|---|---|---|---|
| DTimeMon | Time since extraction of lab sample in minutes | REAL | 0.0 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |

| Parameters | Description | Type | Default |
|---|---|---|---|
| ErrorNum | Output of pending error number. For definition of error numbers by this block, see KalFilt error handling (Page 1286). | INT | -1 |
| ErrorPara | 1= Parameter assignment error | BOOL | 0 |
| EY1...EY7 | Deviation between measured values (Y1...Y7) and prediction of the Kalman filter | REAL | 0.0 |
| EYn1...EYn5 | Deviation between lab results and prediction of the Kalman filter | REAL | 0.0 |
| EY1P_k1...EY1P_k5 | Percent deviation between measured value Y1 and prediction of the Kalman filter for 1 to 5 latest samples | REAL | 0.0 |
| EY2P_k1...EY2P_k5 | Percent deviation between measured value Y2 and prediction of the Kalman filter for 1 to 5 latest samples | REAL | 0.0 |
| EY3P_k1...EY3P_k5 | Percent deviation between measured value Y3 and prediction of the Kalman filter for 1 to 5 latest samples | REAL | 0.0 |
| EY4P_k1...EY4P_k5 | Percent deviation between measured value Y4 and prediction of the Kalman filter for 1 to 5 latest samples | REAL | 0.0 |
| EY5P_k1...EY5P_k5 | Percent deviation between measured value Y5 and prediction of the Kalman filter for 1 to 5 latest samples | REAL | 0.0 |
| EY6P_k1...EY6P_k5 | Percent deviation between measured value Y6 and prediction of the Kalman filter for 1 to 5 latest samples | REAL | 0.0 |
| EY7P_k1...EY7P_k5 | Percent deviation between measured value Y7 and prediction of the Kalman filter for 1 to 5 latest samples | REAL | 0.0 |
| EYPScaleOut | Calculated scaling of the bar representation of estimation error in the standard view of the faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| LabResIncluded | Laboratory results included | BOOL | 0 |
| NumberU | Number of input variables U | INT | 7 |
| NumberX | Number of state variables X | INT | 15 |
| NumberY | Number of output variables Y | INT | 7 |
| NumberYn | Number of lab sample variables Yn | INT | 1 |
| NumberZ | Number of non-measurable output variables Z | INT | 0 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with the settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| P11...P1515 | Diagonal elements of covariance matrix *P*, related to uncertainty of differential equation for respective states X1...X15 | REAL | 0.0 |
| SampExtracted | Laboratory sample extracted | BOOL | 0 |

| Parameters | Description | Type | Default |
|---|---|---|---|
| SaPo_ExtAct | 1 = External lab sampling is active<br>0 = Internal lab sampling is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SaRe_ExtAct | 1 = External lab results active<br>0 = Internal lab results active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Status1 | Status word 1 | DWORD | 16#00000000 |
| Status2 | Status word 2 (not used) | DWORD | 16#00000000 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| X1...X15 | Corrected state estimation value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| U1Out...U7Out | Input variables set at the block input (copy of U1...U7) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Y1Filt...Y7Filt | Process output variables calculated based on the corrected states X1...X15 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Y1Out...Y7Out | Process output variables set at the block input (copy of Y1...Y7) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Yn1Out...Yn5Out | Lab results set at the block input | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Z1Out...Z5Out | Additional non-measurable output variables | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

## See also

Description of KalFilt (Page 1276)
KalFilt modes (Page 1282)
KalFilt messaging (Page 1292)
KalFilt block diagram (Page 1292)

## 6.16.6 KalFilt messaging

### Messaging

This block does not offer messaging.

### See also

Description of KalFilt (Page 1276)
KalFilt modes (Page 1282)
KalFilt functions (Page 1284)
KalFilt error handling (Page 1286)
KalFilt I/Os (Page 1287)
KalFilt block diagram (Page 1292)

## 6.16.7 KalFilt block diagram

### KalFilt block diagram

A block diagram is not provided for this block.

### See also

Description of KalFilt (Page 1276)
KalFilt modes (Page 1282)
KalFilt functions (Page 1284)
KalFilt error handling (Page 1286)
KalFilt I/Os (Page 1287)
KalFilt messaging (Page 1292)

## 6.16.8 Operator control and monitoring

### 6.16.8.1 KalFilt views

#### Views of the KalFilt block

The block KalFilt provides the following views:

- KalFilt standard view (Page 1293)
- KalFilt parameter view (Page 1297)
- KalFilt parameter view 2 (Page 1298)
- KalFilt preview (Page 1299)
- KalFilt measurements view (Page 1301)
- KalFilt trend view (Page 1302)
- Batch view (Page 336)
- Memo view (Page 338)
- Block icon for KalFilt (Page 1302)

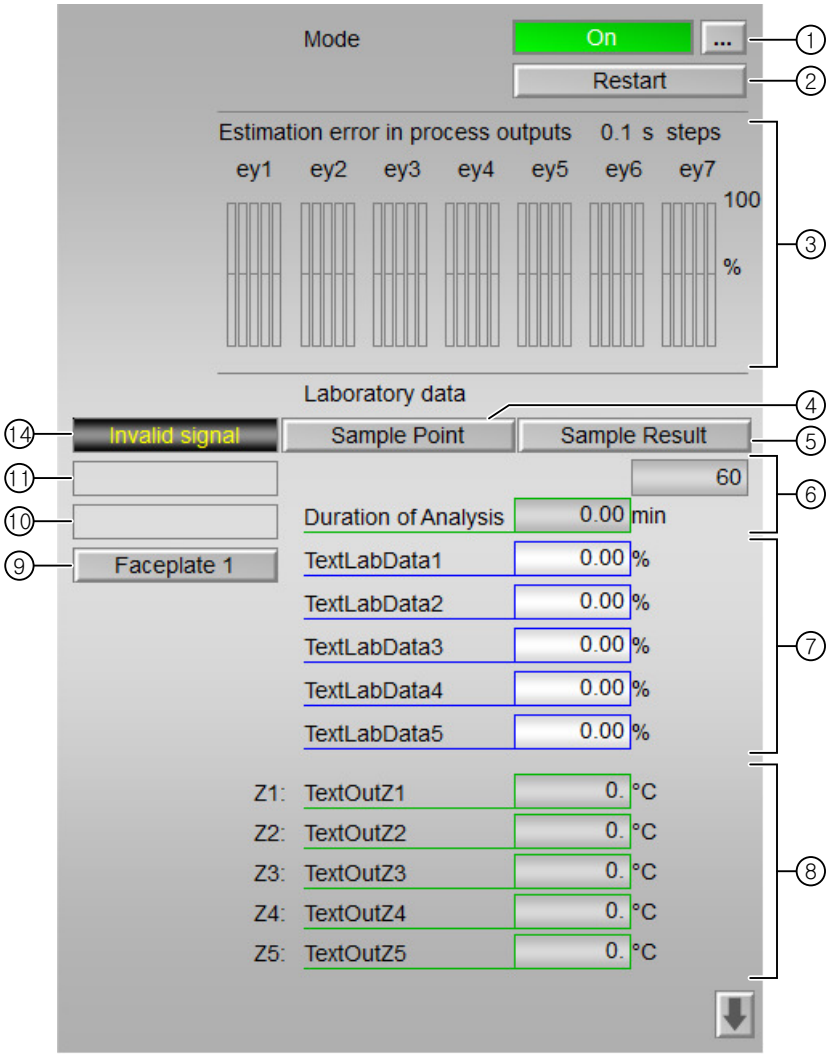
Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 6.16.8.2 KalFilt standard view

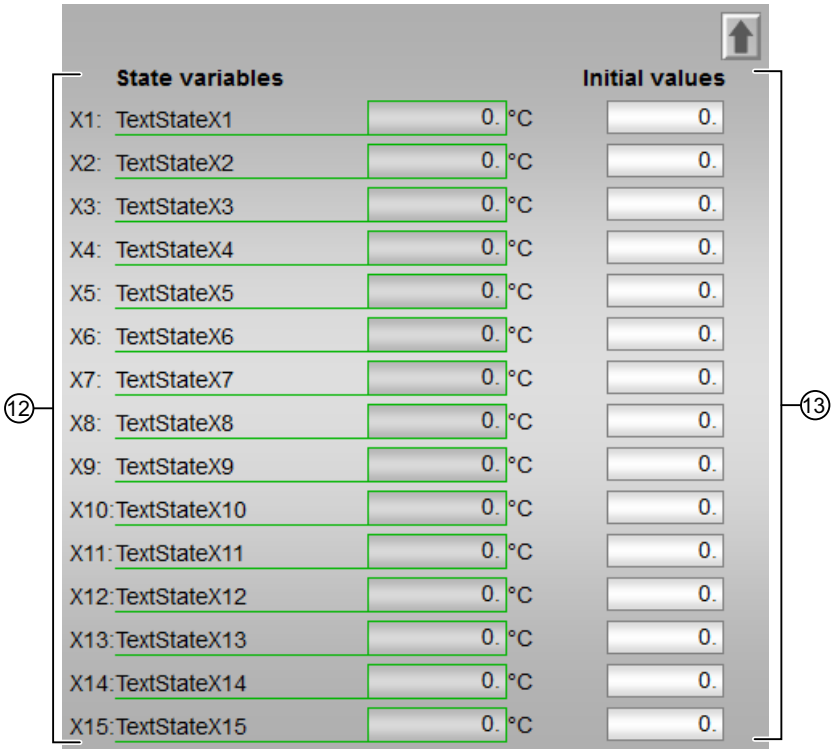
#### Standard view of KalFilt

The standard view has two parts; view 1 and view 2. You can change between the two views with the arrow keys.

**View 1**


Mode
On
...
Restart
Estimation error in process outputs
0.1 s steps
ey1
ey2
ey3
ey4
ey5
ey6
ey7
100
%
Laboratory data
Invalid signal
Sample Point
Sample Result
60
Duration of Analysis
0.00
min
Faceplate 1
TextLabData1
0.00
%
TextLabData2
0.00
%
TextLabData3
0.00
%
TextLabData4
0.00
%
TextLabData5
0.00
%
Z1:
TextOutZ1
0.
°C
Z2:
TextOutZ2
0.
°C
Z3:
TextOutZ3
0.
°C
Z4:
TextOutZ4
0.
°C
Z5:
TextOutZ5
0.
°C
1
2
3
4
5
6
7
8
9
10
11
14

**View 2**




### (1) Displaying and switching the operating mode

This area provides information on the currently active operating mode. The following operating mode can be shown here:

- On (Page 77)
- Out of service (Page 77)

Refer to chapter Switching operating states and operating modes (Page 289) for information on switching the operating mode.

### (2) Restarting the block

Click "Restart" to restart the Kalman filter block.

The initial values specified at the block inputs (`X1Start`...`X15Start`) can be set with `Reset`.

### (3) Display of the estimation error in process outputs

In this area, the percentage estimation errors in the existing measurement channels are displayed. These are based on the particular value range and are calculated for 5 consecutive sampling instants in the recent past. The estimation error at the current time is displayed on the far right and the most distant error in the past (5 sampling steps ago) can be seen at the far left. You can change the scaling of the bar in the parameter view (Page 1297) of the faceplate.

### (4) Sample Point

You can mark the time of extracting lab sample using "Sample Point" button (only if `SaPo_LiOp` = 0).

### (5) Sample Result

Click this button to confirm the laboratory results entered in the "TextLabDatai" fields. Refer to **(7) Laboratory results**.

### (6) Duration of Analysis

This field displays the elapsed time since the extraction of lab sample. If this is greater than the specified limit (`TimeMon`, can be changed in the parameter view), which is displayed at the upper right field, the result of the laboratory measurement is no longer used.

### (7) Laboratory results

You can enter the laboratory results (`Yni_Int`) directly in the "TextLabDatai" fields. Once they are all fully entered, the entry must be confirmed with the "Sample Result" button. The counter stops and the laboratory data are considered in the calculations of the Kalman filter. If a sample is extracted but no meaningful result can be given, a new sample extraction can be started in the faceplate. If the laboratory data is given via interconnection, the description for engineering and operation can be found in the section Additional configuration tasks for inclusion of the laboratory measurements (Page 1276).

### (8) Additional output variables

If you have output variables which cannot be measured and are calculated from *a posteriori* state estimates, you can define and enter the calculation algorithm for a maximum of 5 additional output variables (`Z1Out`...`Z5Out`) in the Kalman Configurator.

### (9) Navigation button for switching to the standard view of any faceplate

Use this navigation button to open the standard view of another function block, which is configured in the engineering system (ES).

The visibility of this navigation button depends on the configuration in the engineering system (ES).

For more information, refer to the section Opening additional faceplates (Page 234).

### (10) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"

You will find more detailed information on this in the sections Simulating signals (Page 64).

### (11) Display area for block states

This area provides additional information on the operating state of the block:

- "Sample extracted"
- "Lab results included"
- "Error in lab results"
- "Time out in analysis"
- "Lab results limited"

### (12) State variables

This area shows the current values of the estimated state variables.

### (13) Initial values

You can specify intial values for the state variables, which are used for `Reset`.

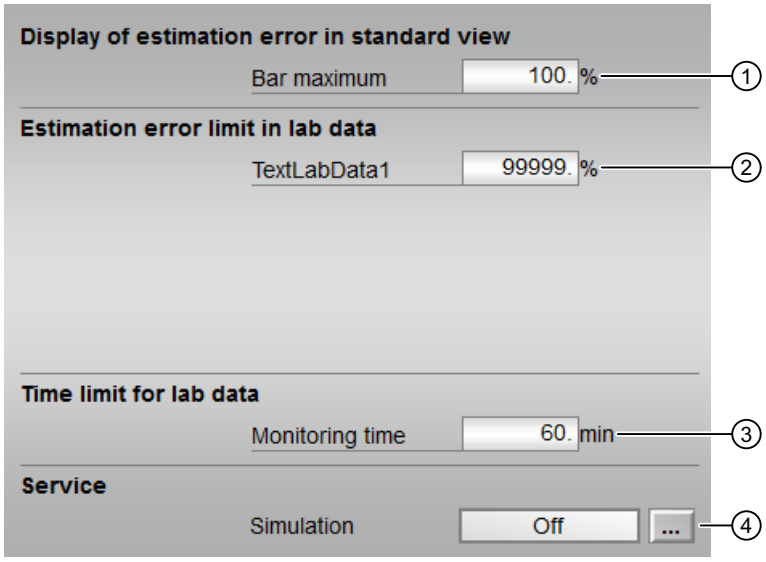
### (14) Display area for block states

This area provides additional information on the operating state of the block:

- "Internal error"
- "Invalid Signal"

## 6.16.8.3 KalFilt parameter view

### Parameter view of KalFilt



### (1) Display of estimation error in standard view

You can set the scale of the bar diagram in the "Bar maximum" (`EYPScale`) field. The bar diagram is displayed in the KalFilt standard view (Page 1293).

### (2) Estimation error limits in lab data

You can change the maximum permissible deviation of the laboratory results from the estimated state variables in the "TextLabData1" field (`EYniLim`).

If the actual error is above this value, the lab results are regarded as implausible and not used for the Kalman filter.

### (3) Duration limits in lab data

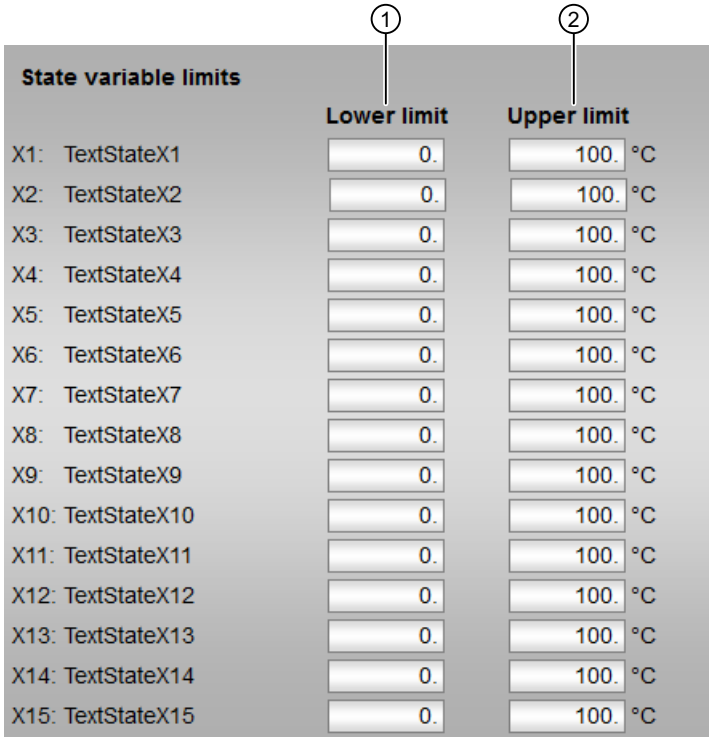
You can adjust the maximum duration for a laboratory measurement in the "Monitoring time" field.

### (4) Service

You can enable and disable simulation of the block using the "Simulation" button.

## 6.16.8.4 KalFilt parameter view 2

### Parameter view 2 of KalFilt



### (1) Lower limit of state variables

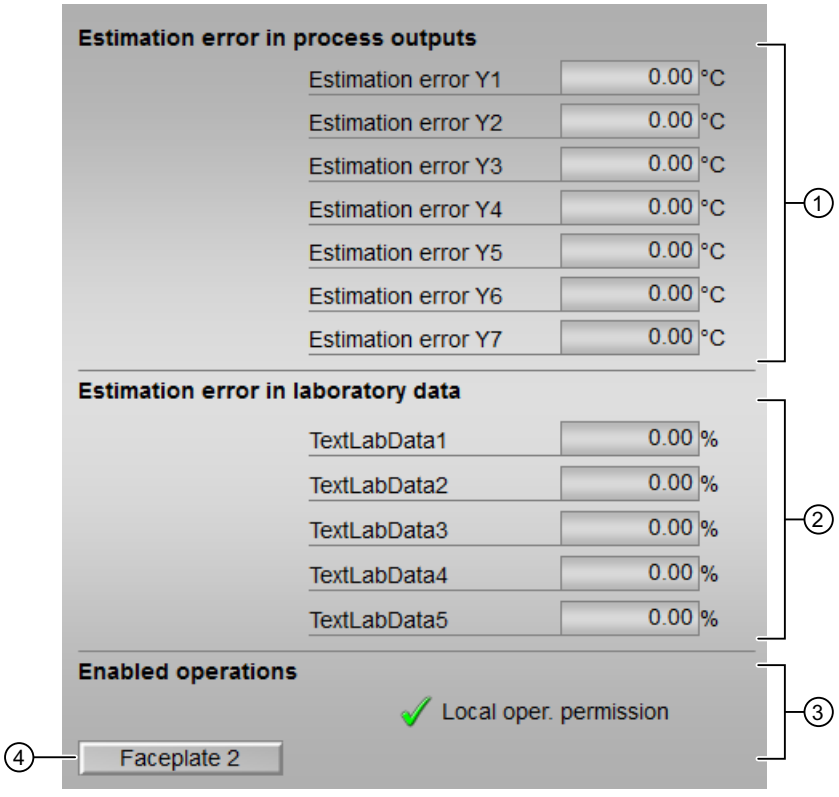
You can specify the lower limit of the state variables.

### (2) Upper limit of state variables

You can specify the upper limit of the state variables.

## 6.16.8.5 KalFilt preview

### Preview of Kalfilt




### (1) Estimation error in process variables

This area shows the estimation error in the output variables. The values of the estimation error are shown in physical units in preview and as percentage in the standard view (Page 1293).

### (2) Estimation error in laboratory data

This area shows the estimation error in the laboratory results. The values of the estimation error are shown in physical units in preview and as percentage in the standard view (Page 1293).

### (3) Enable operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

The following enabled operation is shown here:

- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. You can find additional information on this in the section Operator control permissions (Page 285).

### (4) Navigation button for switching to the standard view of any faceplate

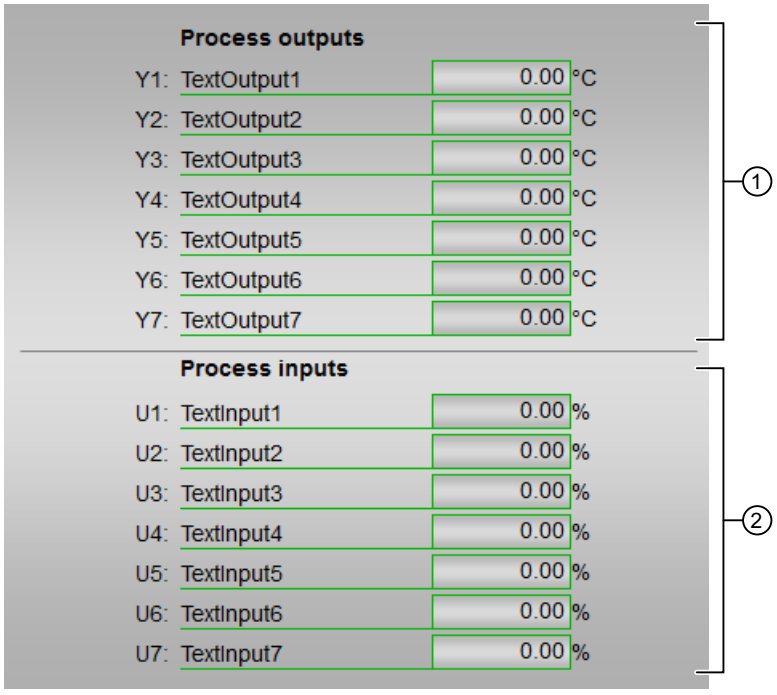
Use this navigation button to open the standard view of another function block, which is configured in the engineering system (ES).

The visibility of this navigation button depends on the configuration in the engineering system (ES).

For more information, refer to the section Opening additional faceplates (Page 234).

### 6.16.8.6 KalFilt measurements view

#### Measurements view of KalFilt




#### (1) Process outputs

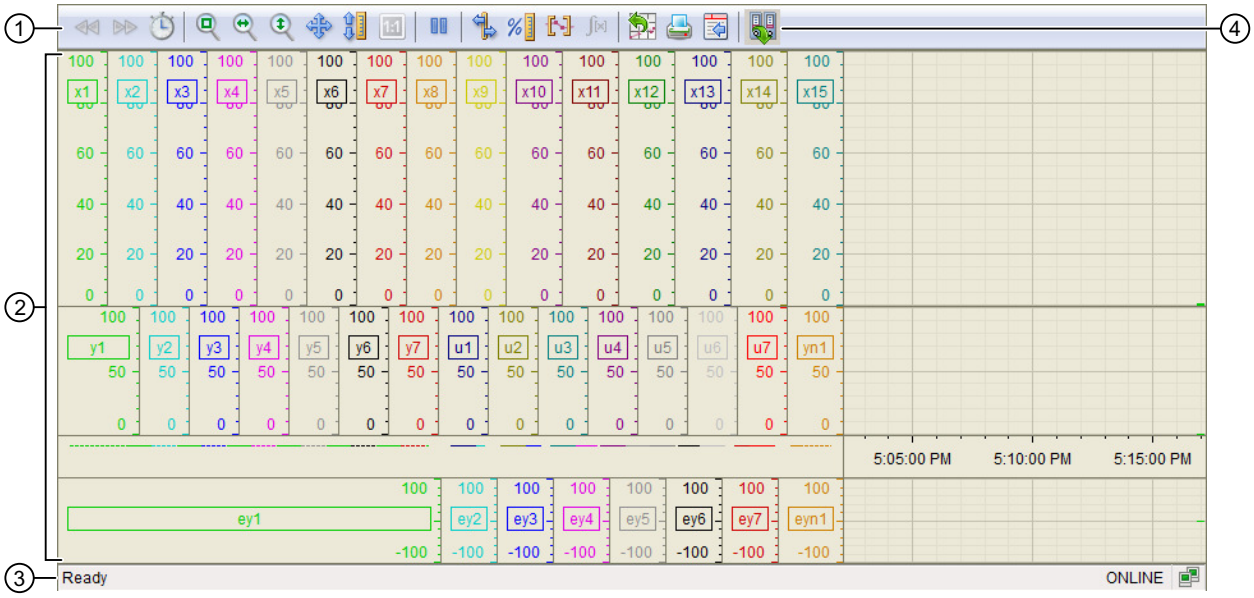
This area displays current values of the process outputs.

#### (2) Process inputs

This area displays current values of the process inputs.

### 6.16.8.7 KalFilt trend view

#### Trend view of KalFilt




**(1)** Toolbar

**(2)** Display area for trends

**(3)** Status bar

**(4)** Button for switching between archive tags and online tags. The status bar shows if the trend view is working with online data or archive data.

For additional information about the trend view, refer to the *WinCC Information System* Online Help.

The trend view is divided into three areas. The top area shows the state variables, the middle area shows the input and output variables, and the lower area shows the estimation errors.

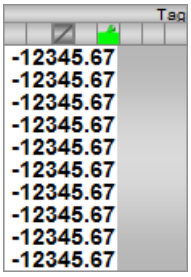
### 6.16.8.8 Block icon for KalFilt

#### Block icons for KalFilt

A single block icon is available with the following functions:

- Process tag type
- Operating modes
- Signal status, release for maintenance
- Memo display
- User analog values 1 to 10, typically a subset of the estimated states and additional output variables (black, with and without decimal places)

The display of the tag name can be activated centrally for all block instances with the WinCC variable @APLShowTag:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon with full display for 10 freely assignable measured values:<br>`UserAna1...UserAna10` |

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 267)
- Operation via the block icon (Page 269)
- Block icon structure (Page 260)

# Dosing blocks

# 7

## 7.1 DoseL - Dosing device

### 7.1.1 Description of DoseL

#### Object name (type + number) and family

Type + number: FB 1809

Family: Dosage

#### Area of application for DoseL

The block is used for the following applications:

- Single-component dosing using flow measurement
- Weighing of fill/removal volume using dosing scales

#### How it works

Processing is performed discretely via a coarse/fine flow control with flow monitoring and setpoint specification. The dosing flow can be determined via a maximum of 16 cycles.

A dribbling phase and post dosing can be implemented for both processes. The input value can also be supplied from a pulse module via the `PV1CycLi` output of the Pcs7Cntx (x=1...3) channel block.

**Note**

When using a pulse module, the counted pulses per cycle must be normalized via the I/Os "Gain" and "Ti" at DoseL .

Example:

A pulse of 35 Kg per hour means that I/O "Gain" must be configured to 35 and I/O "Ti" to 3600.

#### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

For the DoseL block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Dosing (Dose_Lean) (Page 2856)
- Dosing with PA/FF devices (Dose_Lean_Fb) (Page 2857)

### Startup characteristics

Use the `Feature.Bit0` Set startup characteristics (Page 148) to define the startup characteristics of this block and `Feature2.Bit7` Use the last value following a complete download as the current value during startup of the block (Page 165).

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section DoseL I/Os (Page 1332).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `AutoAct.Value` |
| 6 | `LocalAct.Value` |
| 7 | 0: Open padlock in the block icon<br>1: Closed padlock in the block icon |
| 8 | `SP_ExtAct` |
| 9 | `Ctrl` |
| 10 | `Ctrl2` |
| 11 | 1 = Dosing by scale |
| 12 | 0 = Dosing by scale for filling<br>1 = Dosing by scale for removal |
| 13 | `BypProt` active |
| 14 | Invalid signal status |
| 15 | Mode switchover error |
| 16 | 1 = `Intlock` is active |
| 17 | 1 = `Permit` is active |
| 18 | 1 = `Protect` is active |
| 19 | `DosOn` |
| 20 | `DosRelax` |
| 21 | `DosEnd` |
| 22 | `DosOff` |
| 23 | `DosPause` |
| 24 | `DosStart` |
| 25 | 1 = Post dosing |

| Status bit | Parameter |
|---|---|
| 26 | "Start" command |
| 27 | "Pause" command |
| 28 | "Continue" command |
| 29 | "Cancel" command |
| 30 | `UserAna1` interconnected |
| 31 | `UserAna2` interconnected |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `MsgLock` |
| 1 | `DQ_AH_Act xx_Act` taking into account `Feature.Bit29` |
| 2 | `DQ_AL_Act xx_Act` taking into account `Feature.Bit29` |
| 3 | `DQ_AH_En` OR `Feature.Bit16` |
| 4 | `DQ_AL_En` OR `Feature.Bit16` |
| 5 | `DQ_AH_MsgEn` |
| 6 | `DQ_AL_MsgEn` |
| 7 | `PV_AH_Act xx_Act` taking into account `Feature.Bit29` |
| 8 | `PV_AL_Act xx_Act` taking into account `Feature.Bit29` |
| 9 | `PV_AH_En` and `Feature.Bit7` = 1 or `PV_AH_En` and flow mode OR `Feature.Bit16` |
| 10 | `PV_AL_En` and `Feature.Bit7` = 1 or `PV_AL_En` and flow mode OR `Feature.Bit16` |
| 11 | `PV_AH_MsgEn` |
| 12 | `PV_AL_MsgEn` |
| 13 | `PV_AH2_Act xx_Act` taking into account `Feature.Bit29` |
| 14 | `PV_AL2_Act xx_Act` taking into account `Feature.Bit29` |
| 15 | `PV_AH2_En` and `Feature.Bit7` = 1 or `PV_AH2_En` and flow mode OR `Feature.Bit16` |
| 16 | `PV_AL2_En` and `Feature.Bit7` = 1 or `PV_AL2_En` and flow mode OR `Feature.Bit16` |
| 17 | `PV_AH2_MsgEn` |
| 18 | `PV_AL2_MsgEn` |
| 19 | `CR_AH_Act xx_Act` taking into account `Feature.Bit29` |
| 20 | `CR_AH_En` and `Feature.Bit7` = 1 or `CR_AH_En` and flow mode OR `Feature.Bit16` |
| 21 | `CR_AH_MsgEn` |
| 22 | Display for interlocks in block icon |
| 23 | 1 = Scales tared |
| 24 | Automatic preview 1 = Dosing "On" |
| 25 | Automatic preview 1 = Dosing "Dribbling" |
| 26 | Automatic preview 1 = Dosing "End" |
| 27 | Automatic preview 1 = Dosing "Off" |

| Status bit | Parameter |
|---|---|
| 28 | Automatic preview 1 = Dosing "Pause" |
| 29 | Forcing active |
| 30 | Bypass information for display in block icon |
| 31 | `MS_RelOp` |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 | "Interlock" button is enabled |
| 1 | "Permission" button is enabled |
| 2 | "Protection" button is enabled |
| 3 | `DosCancelMsgEn` |
| 4 | `Feature.Bit7` = 1 (Calculation of the flow rate for dosing by scale) |
| 5 | `Feature.Bit7` = 1 or flow mode |
| 6 | Display flow setpoint in percent |
| 7 | Display flow setpoint bar |
| 8 | Reset request in automatic mode |
| 9 | `Feature.Bit8` = 1 (Fine dosing quantity setpoint absolute) |
| 10 | `SimLiOp.Value` |
| 11 | External error generated by FaultExt or external control system fault CSF with set `Feature.Bit18` Activating error state for external process control error CSF (Page 162) |
| 12 | 0 = Button text "Ack underdose" is visible<br>1 = Button text "Ack overdose" is visible |
| 13 | Display "Ack Dos End" block state in the standard view |
| 14 | 1= No impact of input signals on "local mode" with LocalSetting = 0 and OOS mode. |
| 15 - 17 | Not used |
| 18 | Internal error |
| 19 | Display "Forced start" |
| 20 | Display "Force abort" |
| 21 | Display "Forced pause" |
| 22 | Display "Forced continue" |
| 23 - 27 | Not used |
| 28 | `GrpErr.Value` |
| 29 | `RdyToStart.Value` |
| 30 | 1 = Input parameter `Ctrl1ChnST` is interconnected |
| 31 | 1 = Input parameter `Ctrl2ChnST` is interconnected |

### Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 | Effective signal 1 of the message block connected via `EventTsIn` |
| 1 | Effective signal 2 of the message block connected via `EventTsIn` |
| 2 | Effective signal 3 of the message block connected via `EventTsIn` |
| 3 | Effective signal 4 of the message block connected via `EventTsIn` |
| 4 | Effective signal 5 of the message block connected via `EventTsIn` |
| 5 | Effective signal 6 of the message block connected via `EventTsIn` |
| 6 | Effective signal 7 of the message block connected via `EventTsIn` |
| 7 | Effective signal 8 of the message block connected via `EventTsIn` |
| 8 | Delay of the `DQ_AH_Tol` message |
| 9 | Delay of the `DQ_AL_Tol` message |
| 10 | Delay of the `PV_AH_Lim` message |
| 11 | Delay of the `PV_AL_Lim` message |
| 12 | Delay of the `PV_AH2_Lim` message |
| 13 | Delay of the `PV_AL2_Lim` message |
| 14 | Delay of the `CR_AH_Lim` message |
| 15 | Collection of message delays |
| 16 - 22 | Not used |
| 23 | Hidden bypass signal in Permit |
| 24 | Hidden bypass signal in interlock |
| 25 | Hidden bypass signal in Protect |
| 26 | `Feature2.Bit2`: Separate bypass signal |
| 27 - 28 | Not used |
| 29 | 1= Suppressed messages |
| 30 - 31 | Not used |

### Status word allocation for `Status5` parameter

| Status bit | Parameter |
|---|---|
| 0 - 7 | Effective signal 9..16 of the message block connected via `EventTsIn` |
| 8 - 31 | Not used |

### See also

DoseL functions (Page 1312)
DoseL messaging (Page 1329)
DoseL modes (Page 1310)
DoseL error handling (Page 1327)
DoseL block diagram (Page 1348)

## 7.1.2 DoseL modes

### DoseL operating modes

The block can be operated using the following modes:

- Local mode (Page 86)
- Automatic mode (Page 82)
- Manual mode (Page 82)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

### "Local mode"

You can find general information on "Local mode", switching modes and Bumpless switchover in the Local mode (Page 86) section.

Dosing actions you can control in "local mode":

- "Start" (positive edge `StartLocal`)
- "Cancel" (`CancelLocal` = 1)
- "Pause" (`PauseLocal` = 1)
- "Continue" (`ContLocal` = 1)

If you operate the block in "local mode", control will only take place via "local" signals (input parameters `StartLocal` = 1, `CancelLocal` = 1, `PauseLocal` = 1 and `ContLocal` = 1).

**Note**

Unlike the general description, at the `LocalSetting` parameter only the values 0, 1 and 3 can be set, i.e. tracking in "local" mode is not possible with DoseL.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and Bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 82) section.

Dosing actions you can control in "automatic mode":

- "Start" (positive edge `StartAut`)
- "Cancel" (`CancelAut` = 1)
- "Pause" (`PauseAut` = 1)
- "Continue" (`ContAut` = 1)

## "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the section Manual and automatic mode for motors, valves and dosers (Page 82).

Dosing actions you can control in "manual mode":

- "Start" (`StartMan` = 1)
- "Cancel" (`CancelMan` = 1)
- "Pause" (`PauseMan` = 1)
- "Continue" (`ContMan` = 1)

## "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

In "Out of service" mode, if `Feature2.Bit7` =1 then Previous value of Dribbout is retained. Else if `Feature2.Bit7` =0, Dribbout value is set to 0

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit3` =1 and `Feature.Bit3` =0 is as shown below:

| Output Parameters | **`Feature.Bit3` =0** | **`Feature.Bit3` =1** |
|---|---|---|
| DosStart.ST | 16#80 | 16#60 |
| DosEnd.ST | 16#80 | 16#60 |
| DosOn.ST | 16#80 | 16#60 |
| DosRelax.ST | 16#80 | 16#60 |
| DosOff.ST | 16#80 | 16#60 |
| DosPause.ST | 16#80 | 16#60 |
| SP_ExtAct.ST | 16#80 | 16#60 |
| DQ_SP.ST | 16#80 | 16#60 |
| DQ1_SP.ST | 16#80 | 16#60 |
| DQ2_SP.ST | 16#80 | 16#60 |
| DQ_SP_Tol.ST | 16#80 | 16#60 |
| DQ_ExtOut.ST | 16#80 | 16#60 |
| DQ2_ExtOut.ST | 16#80 | 16#60 |
| DQ_Out.ST | 16#80 | 16#60 |
| DQ_AH_Act.ST | 16#XX | 16#60 |
| DQ_AL_Act.ST | 16#XX | 16#60 |
| SP.ST | 16#80 | 16#60 |
| SP1.ST | 16#80 | 16#60 |
| SP2.ST | 16#80 | 16#60 |
| SP_ExtOut.ST | 16#80 | 16#60 |
| SP2_ExtOut.ST | 16#80 | 16#60 |
| SP_ExHiAct.ST | 16#80 | 16#60 |
| SP_ExLoAct.ST | 16#80 | 16#60 |
| SP2_ExHiAct.ST | 16#80 | 16#60 |

| Output Parameters | `Feature.Bit3 =0` | `Feature.Bit3 =1` |
|---|---|---|
| SP2_ExLoAct.ST | 16#80 | 16#60 |
| PV_Out.ST | 16#80 | 16#60 |
| PV_AH_Act.ST | 16#XX | 16#60 |
| PV_AL_Act.ST | 16#XX | 16#60 |
| PV_AH2_Act.ST | 16#XX | 16#60 |
| PV_AL2_Act.ST | 16#XX | 16#60 |
| CR_AH_Act.ST | 16#XX | 16#60 |

### See also

Description of DoseL (Page 1305)

DoseL functions (Page 1312)

DoseL error handling (Page 1327)

DoseL messaging (Page 1329)

DoseL I/Os (Page 1332)

DoseL block diagram (Page 1348)

## 7.1.3 DoseL functions

### Functions of DoseL

The functions for this block are listed below.

### Status diagram

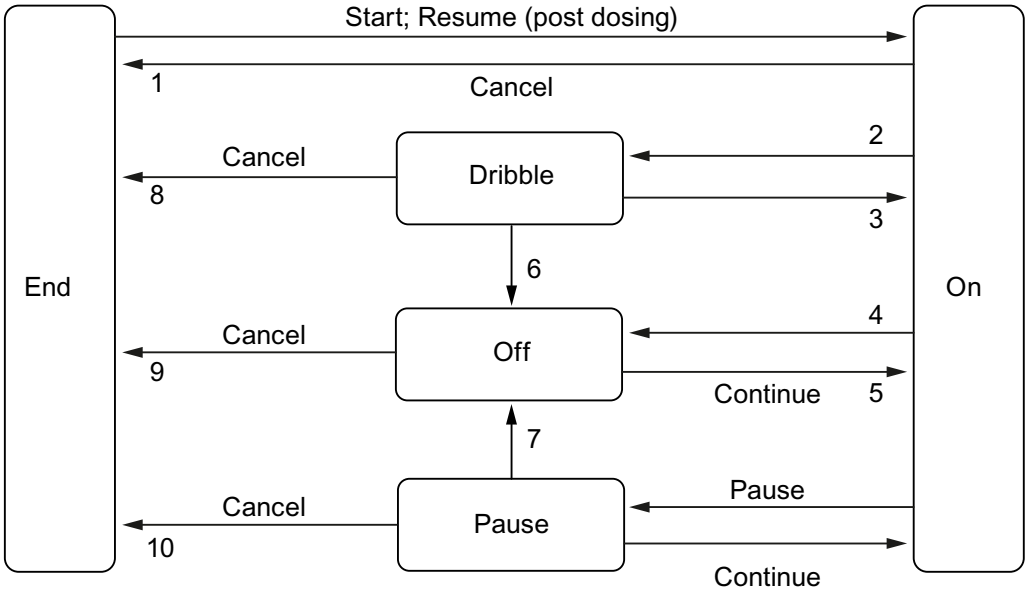
The block provides the following states:

- "End"
  If an underdosage or overdosage is identified, the status display shows the state "Ack Dos End" in the standard view.
- "On"
- "Dribbling"
- "Off"
- "Pause"

The following commands can initiate a state change:

- "Start"
- "Cancel"

- "Pause"
- "Continue"




Dosing can only be started if the dosing setpoint is greater than the current dosing quantity (`DQ_Out`) or if `Feature.Bit6` is set and no interlock is pending.

**Note**

After a start of scale dosing with `Feature.Bit14` = 1 the controls are deactivated during the scale is in taring mode (`StandStill` = 0).

However, the following status changes are also performed automatically:

| Number in graphic (top) | Function |
|---|---|
| 1 | When the dosing quantity is reached, dosing terminates (`DQ_Out` ≥ DQ_SP). |
| 2 | If the dosing quantity reaches the dribbling quantity (relative to dosing setpoint `DQ_Out` ≥ `DQ_SP` – `DribbOut`) the dribbling status becomes active. |
| 3 | Automatic post dosing in automatic mode via the `Feature.Bit12`, if an under-dosage was identified after the dribbling time (`RelaxTime`). |
| 4 | • Flow alarm (see input parameter `Feature.Bit11`) or<br>• Interlock.<br>• The dosing process of the DoseL block is set to the "Off" state with a process value status "Bad, device-related" or "Bad, process-related". The response can be set using `Feature` bit parameter assignment. |
| 5 | • The flow alarm is acknowledged (also provided via the "Continue" command if `Feature.Bit9` = 1) or<br>• The interlock is acknowledged (also provided via the "Continue" command if `Feature.Bit9` = 1).<br>• "Continue" command after underdosing. |

| Number in graphic (top) | Function |
|---|---|
| 6 | Underdosage identified after the dribbling time (`RelaxTime`).<br>Overdosage identified after the dribbling time (only with `Feature2.Bit24` = 1). |
| 7 | Interlock |
| 8 | Dribbling time (`RelaxTime`) has expired and dosing quantity is above the low tolerance limit (`DQ_Out` ≥ `DQ_SP_Tol` - `DQ_AL_Tol`) |
| 9 | The underdosage is acknowledged or the dosing quantity has been reached (`DQ_Out` ≥ `DQ_SP`) via creep flow, for example |
| 10 | Dosing quantity has been reached (`DQ_Out` ≥ `DQ_SP`) via creep flow, for example |

### Control outputs

The coarse flow (`Ctrl`) or fine flow (`Ctrl2`) control outputs are issued in the "On" status.

The coarse flow control output is active if:

- `DQ_Out` < `DQ_SP` – `DQ2_SP`

The fine flow control output is active if:

- `DQ_Out` ≥ `DQ_SP` – `DQ2_SP`
- Where `DQ_Out`: Dosing quantity actual value
- `DQ_SP`: Dosing quantity setpoint
- `DQ2_SP`: fine dosing quantity setpoint ("calculated")

### Output signal as a pulse signal or static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 57).

In addition to the static control outputs `Ctrl`, `Ctrl2`, the block also has pulse outputs `P_Ctrl`, `P_Ctrl2`, which are dependent on the static control outputs.

### Determining the dosing quantity when using flow dosing

You can activate flow dosing using `Feature.Bit5` = 0 (Specifying the dosing type (Page 157).)

When using flow dosing, the dosing quantity is determined using the following equation:

$$DQ_Out = DQ_Out + \frac{SampleTime}{TI} \cdot \frac{(PV_Out_{alt} + PV_Out)}{2}$$

Where:

- `DQ_Out`: Dosing quantity actual value
- `SampleTime`: Sampling time [s]
- `TI`: dimensionless conversion factor for the time basis of the measured value acquisition in the time basis 1 s
- $PV_Out_{alt}$: Last value `PV_OUT`

The output parameter PV_OUT is determined using an average calculation:

$$PV_Out := \frac{PV_Out_{(t)} + PV_Out_{(t-1)} + ... + PV_Out_{(t-(n-1))}}{NumSample_{(N)}}$$

Where:

- PV_Out(t) = PV · Gain

The average calculation is only active if NumSample > 1 and NumSample ≤ 16 and serves to smooth systematically pulsating measuring signals.

**Note**

**Integration of the flow for the dosing quantity**

The integration of the flow for the dosing quantity is implemented using the trapezoid rule. Compared to the summation of rectangles rule, in which the values are simply added up, the procedural error in the case of the trapezoid rule is smaller for the determination of the numerical integral value.

The dosing quantity is determined in the "On" and "Dribbling" state. In the "End", "Off" and "Pause" states, the flow (creep flow) is determined depending on the Feature.Bit13 (Creep rate is always detected in the dosing quantity (Page 179)) and the value CR_AH_Lim. Creep flow monitoring is disabled with CR_AH_En = 0 .

### Determining the dosing quantity when dosing using scales

You can activate scale dosing via the Feature.Bit5 = 1 (Specifying the dosing type (Page 157).)

The dosing quantity is determined for a scale dosing in the "Start" state following a positive edge at the input parameter StandStill. The StandStill input parameter is a feedback signal of the scale.

The process of determining the dosing quantity stops after the dosing quantity is reset in the "End" state.

If the signal is no longer available, you need to configure StandStill with 1 permanently; the dosing quantity will then be determined right at the start of the dosing.

When weighing the fill volume (MeterType = 0), the dosing quantity is determined using the following equations:

$$DQ_Out = PV_Out - DQ_Tare$$

When weighing the removal volume (MeterType = 1):

$$DQ_Out = DQ_Tare - PV_Out$$

with PV_Out = PV · Gain

In the "Start" state, the `DQ_Tare` tare memory is set with `PV_Out` with the first positive edge of `StandStill`. With `Feature.Bit14` = 1 the dosing controls `Ctrl`, `P_Ctrl`, `Ctrl2`, `P_Ctrl2` are deactivated until the tare memory is set.

If you have permanently configured `StandStill` with 1, the tare memory is set right at the start of dosing.

**Note**

The `DQ_Out` output parameter is the dosing quantity and not the actual scale value. In the "End" state (= following resetting of the dosing quantity), nothing further is displayed, since the dosing procedure is complete. When dosing starts, the actual scale value is transferred to the tare memory, and therefore the dosing quantity always remains 0 after the start of dosing. Therefore, the actual scale value is not displayed in the "End" state.

## Calculation of the flow rate for dosing by scale

You can activate the calculation of the flow rate via the `Feature.Bit7` = 1

(Activating calculation of the flow rate for dosing by scale (Page 155)).

The flow rate for scale dosing is determined by the following equations:

$$PV_Out = (avPV(t) - avPV(t-1)) \cdot \frac{GainEff}{SampleTime}$$

with

$$avPV(t) = \frac{PV(t) + PV(t-1) + ... + PV(t-(N+1))}{NumSample(N)}$$

where `avPV(t)` = internal variable for the mean dosing quantity

## Dribbling

The "Dribbling" status is entered automatically in accordance with

- `DQ_Out` ≥ `DQ_SP` − `DribbOut`

Where:

- `DQ_Out`: Dosing quantity actual value
- `DQ_SP`: Dosing quantity setpoint
- `DribbOut`: Dribbling quantity

The dribbling quantity is specified with the input `DribbIn`.

`DribCor` = 1 can be used to automatically determine the dribbling quantity from previous dosing actions:

- `DribbOut` = `DribbOut` - (`DQ_SP` - `DQ_Out`) · `DCF` / 100

DCF represents the weighting factor of the last dosing action as a percentage and cannot be set to below 0 or above 100. DribbOut is calculated at the end of dosing or the first time an underdosage occurs, and is limited to DribbMax. Post dosing is not taken into account.

The "Dribbling" status is active for the period RelaxTime.

**Note**

Also with DribbOut = 0 the "Dribbling" status is active. If you want to deactivate the "Dribbling" status, you have to set additionally RelaxTime = 0.

When the "Enable configuration of the dribbling quantity (Page 170)" function is enabled (Feature.Bit27 = 1) and automatic determination of the dribbling quantity is disabled (DribCor = 0), the quantity can also be configured during dosing, but not in the "Dribbling" state.

After a complete download you can use the dribbling quantity from the input OldDribbOut with parameterizing Feature2.Bit7 =1.

## Overdosing/underdosing

Once the "Dribbling" state has expired, the dosing quantity is checked for overdosages or underdosages. In the "End" status, no check is performed for over- or underdosages.

An overdosage has occurred if:

- DQ_Out > DQ_SP_Tol + DQ_AH_Tol

An underdosage has occurred if:

- DQ_Out < DQ_SP_Tol - DQ_AL_Tol
- Where DQ_Out: Dosing quantity actual value
- DQ_SP_Tol: Dosing quantity setpoint for forming tolerance. This is the same as the dosing quantity setpoint (DQ_SP) before and after the check that determines overdosing/ underdosing.
- DQ_AH_Tol: High tolerance limit
- DQ_AL_Tol: Low tolerance limit

If an underdosage is identified after the "Dribbling" state, the "Off" state is entered. The underdosage can be acknowledged in "Off" state (U_AckOp = 1 or positive edge U_AckLi) and the dosing action completed, or post dosing can be started with the "Continue" command. The dosing end state display in the standard view shows "Ack Dos End".

**With** Feature2.Bit24 = 0**:**

If an overdosage is identified after the "Dribbling" state, the dosing "End" state is entered at once. A message for overdosage comes and goes in next cycle.

**With** Feature2.Bit24 = 1**:**

If an overdosage is identified after the "Dribbling" state, the "Off" state is entered. The overdosage can be acknowledged in the "Off" state (U_AckOp = 1 or positive edge U_AckLi) and the dosing action is completed.

Generally, dosages which enter the "Off" state with overdosage, have to be acknowledged.

### Post dosing

If an underdosage is detected after dribbling, post dosing can be performed using the "Continue" command. In automatic mode, `Feature.Bit12` can be used to start post dosing automatically. Post dosing is active for the period `P_DoseTime`. The block then enters the "Dribbling" state or, if that state is deactivated, the "End" state. If the conditions for the "Dribbling" or "End" states are met within the period `P_DoseTime`, the block switches to these states immediately.

Post dosing can also be carried out in the "End" state by increasing the dosing quantity setpoint and issuing the "Continue" command. Once the dosing quantity has been reset, post dosing can no longer be carried out by means of increasing the setpoint.

If the dribbling correction quantity is smaller than the dribbling quantity, dosing is started for time `P_DoseTime`. If no parameter assignment has been made for `P_DoseTime`, the coarse flow (`Ctrl`) or fine flow (`Ctrl2`) control outputs are set for one cycle.

If no parameter assignment has been made for the dribbling quantity, or the dribbling correction quantity is larger than the dribbling quantity, the dosing procedure is continued without taking time `P_DoseTime` into account.

Information on post dosing via setpoint increase:

- After the setpoint has been increased, the tolerance limits are only updated when the "Continue" command is issued. This ensures the overdosage/underdosage display remains consistent.
- Post dosing can only be carried out after a dosing procedure has been aborted if the setpoint for the dosing quantity has been increased and an actual-value quantity is present.
- In "Automatic" and "Local" modes, the "Continue" command must be issued on a positive edge to prevent unwanted dosing procedures from taking place after a setpoint increase.

**With** `Feature2.Bit24` = 0**:**

If an overdosage is identified during or after a post dosing, the dosage enters in the "End" state.

**With** `Feature2.Bit24` = 1**:**

If an overdosage is identified after a post dosing or a drain, the dosage continues to be in the "Off" state. Only with acknowledge overdosage, the block enters to the "End" state with the status display "Ack Dos End" in the standard view.

### Resetting the dosing quantity

The dosing quantity can only be reset in the "End" state using `RstDQ_Op` = 1 or positive edge `RstDQ_Li`.

The state display "Ack Dos End" is also resetted to "End".

`Feature.Bit6` can be used to reset the dosing quantity automatically when dosing starts.

### External/internal setpoint specification

This block provides the standard function Setpoint specification - internal/external (Page 136).

The block always requires the setpoint for the dosing quantity (dosing setpoint). All I/Os for the dosing setpoint start with `DQ_...`. The dosing setpoint is comprised of the coarse and fine setpoints for the coarse/fine flow control. All I/Os for the fine setpoint start with `DQ2_...`. The

coarse setpoint is generated internally from the dosing and fine setpoints and is displayed at the output with DQ1_SP. If no fine setpoint is available, the coarse setpoint is equal to the dosing setpoint.

Flow setpoints can be specified for the coarse/fine flow setpoint specification. All I/Os for the coarse flow start with SP_..., and all I/Os for the fine flow with SP2_.... The setpoint for coarse or fine flow is displayed at the output SP in the "On" status, depending on the coarse/fine flow control.

### Setpoint limitation

The setpoints are limited in the block using shared parameters:

- Coarse metering volume: DQ_HiLim, DQ_LoLim
- Fine metering volume: DQ2_HiLim, DQ2_LoLim
- Coarse flow rate: SP_HiLim, SP_LoLim
- Fine flow rate: SP2_HiLim, SP2_LoLim

If the setpoint limit is violated, external setpoints are limited to the limit you specified. If there are internal setpoints, the last valid value is output.

### Bumpless switchover from external to internal setpoint

The parameter SP_TrkExt = 1 is used so that the internal setpoint tracks the external setpoint to achieve a bumpless switchover from the external to the internal setpoint. This allows unwanted jumps at the output parameter to be avoided.

### Limit monitoring of the process value

This block provides the standard function Limit monitoring of the process value (Page 93). The calculated flow PV_Out is monitored in terms of the limit pairs PV_AH_Lim / PV_AL_Lim (coarse flow) or PV_AH2_Lim / PV_AL2_Lim (fine flow), depending on the coarse/fine flow control. Monitoring is only active in the "On" status.

The calculated flow PV_Out is monitored in terms of the limit pairs PV_AH_Lim / PV_AL_Lim (coarse flow) or PV_AH2_Lim / PV_AL2_Lim (fine flow), depending on the coarse/fine flow control. Monitoring is only active in the "On" status.

### Providing limits at the output

For further connections to other blocks the following input parameters will also be displayed on the output:

DQ_AL_TolOut := DQ_AL_Tol;

DQ_AH_TolOut := DQ_AH_Tol;

### Group display SumMsgAct for limit monitoring, CSF and ExtMsgx

The block provides the standard function Group display for limit monitoring, CSF and ExtMsgx (Page 92).

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Forcing operating modes

This block provides the standard function Forcing operating modes (Page 45). The inputs `StartForce`, `CancelForce`, `PauseForce` and `ContForce` force the block into the states "On", "End" or "Pause".

The "Pause" status can only be forced if the block is in the "On" status.

The "On" state can only be forced from the states

- "Off"
- "Pause"
- "End" if the quantity actual valve is less than the quantity setpoint.

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

The flow is simulated with `SimPV`. This input is not active in scale mode, even when the flow is formed by changing quantities (`Feature.Bit7` = 1)

You can simulate the following values:

- Flow dosing: The flow (`PV`) is specified in the setpoint view with the input (`SimPV`, `SimPV_Li`).
- Dosing using scales: The dosing quantity (`DQ_Out`) is specified in the standard view with the input (`SimDQ`, `SimDQ_Li`).

### Interlocks

This block provides the following interlocks:

- Activation enable
- Interlock without reset ("Interlock")
- Interlock with reset ("Protection")

Refer to the section Interlocks (Page 107).

Dosing can only be started if no interlock is present. The interlock function is not active in the "Dribbling" status; nor is the activation enable active in the "On" status. An interlock in the "On" or "Pause" states will cause the block to enter the "Off" status.

### Disabling interlocks

This block provides the standard function Disabling interlocks (Page 111).

### Resetting the block in case of interlocks or errors

This block provides the standard function Resetting the block in case of interlocks or errors (Page 47).

### External error (**FaultExt**), external control system fault (**CSF**)

This block provides the possibility to pass an external error via the FaultExt parameter or an external control system fault via the CSF parameter. See DoseL error handling (Page 1327)

### Group error

This block provides the standard function Outputting group errors (Page 130).

The following parameters are taken into consideration when forming the group error:

- CSF
- FaultExt

### Outputting a signal for start readiness

This block provides the standard function Outputting a signal for start readiness (Page 59).

### Forming the group status for interlocks

This block provides the standard function Forming the group status for interlock information (Page 113).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status ST_Worst for the block is formed from the following parameters:

- PV_Out.ST
- DQ_Out.ST
- DQ_SP.ST
- SP.ST
- Standstill.ST
- LocalLi.ST
- StartLocal.ST
- CancelLocal.ST
- PauseLocal.ST
- ContLocal.ST
- CtrlChnST

- Ctrl2ChnST
- StartAut.ST (only if Feature2.Bit10 = 1)
- CancelAut.ST (only if Feature2.Bit10 = 1)
- PauseAut.ST (only if Feature2.Bit10 = 1)
- ContAut.ST (only if Feature2.Bit10 = 1)
- SP_Ext.ST (only if Feature2.Bit10 = 1)

The following signal status is formed by:

| Signal status | Used status |
|---|---|
| DQ_Out.ST | Formed by the signal status PV_Out.ST. The last DQ_Out.St signal status is also included for flow dosing and the DQ_Tare.ST signal status is included for scale dosing. The worst signal status is applied. The signal status is reset with reset of the dosing quantity. |
| DQ_SP.ST | With an external setpoint DQ_Ext.ST (for DQ_HiLim.ST or DQ_LoLim.ST limit), otherwise good status |
| SP.ST | With an external setpoint SP_Ext.ST (coarse flow) / SP2_Ext.ST (fine flow) (with SP_HiLim.ST / SP_LoLim.ST or SP2_HiLim.ST / SP2_LoLim.ST limit), otherwise good status |

### Considering bad quality of automatic commands or external values

If the Feature2 bit Considering bad quality of automatic commands or external values (Page 200) is set to 1 and one of the following parameters has bad signal status (16#00 or 16#28), the block goes to the neutral position in the "Automatic" mode:

- StartAut.ST
- CancelAut.ST
- PauseAut.ST
- ContAut.ST
- SP_Ext.ST

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Alarm delays with two time values per limit pair

This block includes the standard function alarm delay for Two time values per limit pair (Page 227).

### Display and operator input area for process values and setpoints

This block provides the standard function Display and operator input area for process values and setpoints (Page 233).

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 230).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 3 | Enabling resetting of commands for the control settings (Page 172) |
| 4 | Setting switch or button mode (Page 177) |
| 5 | Specifying the dosing type (Page 157) |
| 6 | Resetting the dosing quantity when dosing starts (Page 173) |
| 7 | Activating calculation of the flow rate for dosing by scale (Page 155) |
| 8 | Use an internal or external setpoint for the absolute fine dosing quantity (Page 164) |
| 9 | Resetting via input signals in the event of interlocking (Protection) or errors (Page 174) |
| 10 | Exiting local mode (Page 189) |
| 11 | Stopping dosing at a flow alarm (Page 147) |
| 12 | Automatic post dosing for underdosing in automatic mode (Page 153) |
| 13 | Creep rate is always detected in the dosing quantity (Page 179) |
| 14 | Dosing controls are deactivated during scale is in taring mode (Page 205) |
| 15 | Flow setpoints in percent (Page 157) |
| 16 | Limits disabled in faceplate (Page 206) |
| 17 | Enabling bumpless switchover to automatic mode for valves, motors, and dosers (Page 184) |
| 18 | Activating error state for external process control error CSF (Page 162) |
| 21 | Enable bumpless switchover to "Automatic" mode for operator only (Page 183) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 23 | Specifying the influence of the signal status on the dosing process (Page 158) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 27 | Enable configuration of the dribbling quantity (Page 170) |
| 28 | Disabling operating points (Page 156) |

| Bit | Function |
|---|---|
| 29 | Signaling limit violation (Page 181) |
| 30 | Set reset depending on the operating mode or the LiOp parameter (Page 174) |
| 31 | Activating reset of protection / error in manual mode (Page 175) |

Comments on table:

**`Feature.Bit4`:**

- Button mode (`Feature.Bit4` = 0): The automatic commands in automatic mode are pulse signals, in other words `StartAut`, `CancelAut`, `PauseAut`, `ContAut` can be reset to 0 after switchover to the selected status. In "manual" and "local" modes, however, the automatic commands are static signals, the block state is tracked in the absence of automatic commands.
- Switch mode (`Feature.Bit4` = 1): The conditions are selected via the inputs `StartAut` (0: Cancel, 1: Start) and `PauseAut` (0: Continue, 1: Pause) with static signals.

**`Feature.Bit23`:**

**Note**

When there is a bad signal status and `Feature.Bit23` = 1 , the flow is no longer recorded in the dosing quantity in any state. The configure creep rate alarm is output.

### Configurable reactions using the `Feature2` parameter

| Bit | Function |
|---|---|
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 3 | Control priority in the event of invalid or not plausible input commands (Page 186) |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) |
| 6 | Changes signal status of outputs in OOS (Page 208) |
| 7 | Use the last value following a complete download as the current value during startup of the block (Page 165) |
| 8 | Forcing commands in the Local mode (Page 198) |
| 10 | Considering bad quality of automatic commands or external values (Page 200) |
| 11 | Suppress MsgLock and “Out of service” mode for a connected message block (Page 202) |
| 12 | Disable limit act output when limit value message is disabled (Page 209) |
| 24 | With acknowledge overdosage (Page 190) |

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" |
| 1 | 1 = Operator can switch to "manual mode" |
| 2 | 1 = Operator can switch to "local mode" |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can start dosing |
| 5 | 1 = Operator can pause dosing |
| 6 | 1 = Operator can continue dosing |
| 7 | 1 = Operator can cancel dosing |
| 8 | 1 = Operator can reset interlock and flow alarms |
| 9 | 1 = Operator can reset dosing quantity |
| 10 | 1 = Operator can acknowledge overdosage and underdosage |
| 11 | 1 = Operator can select external setpoints |
| 12 | 1 = Operator can select internal setpoints |
| 13 | 1 = Operator can enter dosing setpoint |
| 14 | 1 = Operator can enter dosing setpoint factor for fine dosing |
| 15 | 1 = Operator can enter flow setpoint for coarse flow |
| 16 | 1 = Operator can enter flow setpoint factor for fine flow |
| 17 | 1 = Operator can activate external/internal bumpless switchover |
| 18 | 1 = Operator can enable simulation function |
| 19 | 1 = Operator can activate the Release for maintenance function |
| 20 | Not used |
| 21 | 1 = Operator can enter dribbling quantity |
| 22 | 1 = Operator can enter maximum dribbling quantity |
| 23 | 1 = Operator can activate/deactivate automatic generation of dribbling quantity |
| 24 | 1 = Operator can enter weighting factor for generation of dribbling quantity |
| 25 | 1 = Operator can change the simulation value SimPV |
| 26 | 1 = Operator can change the simulation value SimDQ |
| 27 - 31 | Not used |

The block has the following operator permissions for the `OS1Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can change the limit for the high alarm for `PV_OUT` (coarse flow) |
| 1 | 1 = Operator can change the limit for the low alarm for `PV_OUT` (coarse flow) |
| 2 | 1 = Operator can change the limit for the hysteresis `PV_OUT` (coarse flow) |
| 3 | 1 = Operator can change the limit for the high alarm for `PV_OUT` (fine flow) |
| 4 | 1 = Operator can change the limit for the low alarm for `PV_OUT` (fine flow) |
| 5 | 1 = Operator can change the limit for the hysteresis `PV_OUT` (fine flow) |
| 6 | 1 = Operator can change the operation high limit of the dosing setpoint |
| 7 | 1 = Operator can change the operation low limit of the dosing setpoint |
| 8 | 1 = Operator can change operation high limit of the dosing setpoint factor for fine dosing |
| 9 | 1 = Operator can change operation low limit of the dosing setpoint factor for fine dosing |

| Bit | Function |
|---|---|
| 10 | 1 = Operator can change the tolerance value for overdosage |
| 11 | 1 = Operator can change the tolerance value for Underdosage |
| 12 | 1 = Operator can change operation high limit of the flow setpoint for coarse flow |
| 13 | 1 = Operator can change operation low limit of the flow setpoint for coarse flow |
| 14 | 1 = Operator can change operation high limit of the flow setpoint for fine flow |
| 15 | 1 = Operator can change operation low limit of the flow setpoint for fine flow |
| 16 | 1 = Operator can change the dribbling time |
| 17 | 1 = Operator can change the post dosing |
| 18 | 1 = Operator can change the limit for the high alarm for `PV_OUT` (creep flow) |
| 19 | 1 = Operator can change the limit for the hysteresis for `PV_OUT` (creep flow) |
| 20 | 1 = Operator can activate / deactivate messages via `DQ_AH_MsgEn` |
| 21 | 1 = Operator can activate / deactivate messages via `DQ_AL_MsgEn` |
| 22 | 1 = Operator can activate / deactivate messages via `PV_AH_MsgEn` |
| 23 | 1 = Operator can activate / deactivate messages via `PV_AL_MsgEn` |
| 24 | 1 = Operator can activate / deactivate messages via `PV_AH2_MsgEn` |
| 25 | 1 = Operator can activate / deactivate messages via `PV_AL2_MsgEn` |
| 26 | 1 = Operator can activate / deactivate messages via `CR_AH_MsgEn` |
| 27 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 239).

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of DoseL (Page 1305)

DoseL messaging (Page 1329)

DoseL I/Os (Page 1332)

DoseL modes (Page 1310)

DoseL block diagram (Page 1348)

EventTs functions (Page 2046)

## 7.1.4 DoseL error handling

### Error handling of DoseL

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Mode switchover error
- Control system fault (CSF)

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. |
| 11 | Configuration error `TI` ≤ 0 |
| 12 | Configuration error:<br>`PV_AH_Lim` < `PV_AL_Lim`<br>`PV_AH2_Lim` ≤ `PV_AL2_Lim`<br>`DQ_HiLim.Value` ≤ `DQ_LoLim.Value`<br>`DQ2_HiLim.Value` ≤ `DQ2_LoLim.Value`<br>`SP_HiLim.Value` ≤ `SP_LoLim.Value`<br>`SP2_HiLim.Value` ≤ `SP2_LoLim.Value` |
| 30 | `PV` is not a valid number |
| 31 | `PV_Out` is not a valid number |
| 32 | `DQ_Out` is not a valid number |
| 41 | The value for the `LocalSetting` I/O is not within the approved limit of 0, 1 or 3. |
| 42 | `LocalSetting` = 0 and `LocalLi` = 1 |
| 48 | `SP_LiOp` = 1 and `SP_IntLi` = 1 and `SP_ExtLi` = 1 |
| 51 | `AutModLi` = 1 and `ManModLi` = 1 and `Feature` Bit Setting switch or button mode (Page 177) = 0<br>or two I/Os are set at the same time:<br>• in "local mode" `StartLocal` (pos. edge), `CancelLocal`, `PauseLocal`, `ContLocal`when forcing states: `StartForce`, `CancelForce`, `PauseForce`, `ContForce`in "automatic mode" with pushbutton operation: `StartAut` (pos. edge), `CancelAut`, `PauseAut`, `ContAut`<br>• in "manual mode": `StartMan`, `CancelMan`, `PauseMan`, `ContMan` |

### Mode switchover error

This error can be output by the block, see the section Error handling (Page 128).

### Control system fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 128) section for more on this.

### Invalid input signal

These errors can be output by the block, see the section Error handling (Page 128).

For the following invalid input signals, the control output can be kept or switched to the neutral position. This depends on the function Control priority in the event of invalid or not plausible input commands (Page 186).

| **Operating mode** | **Invalid input signal** | **Control reaction with** `Feature2.Bit3` = 1 |
|---|---|---|
| Local: `Localsetting` = 1 or `Localsetting` = 3 | `StartLocal` = 1 (pos. edge) and `CancelLocal` = 1 | Dosing canceled |
| | `ContLocal` = 1 and `PauseLocal` = 1 | Dosing paused |
| Local: `Localsetting` = 1 or `Localsetting` = 3 and forcing | `StartForce` = 1 and `CancelForce` = 1 | Dosing canceled |
| | `ContForce` = 1 and `PauseForce` = 1 | Dosing paused |
| Forcing and no "local mode" | `StartForce` = 1 and `CancelForce` = 1 | Dosing canceled |
| | `ContForce` = 1 and `PauseForce` = 1 | Dosing paused |
| "Automatic mode" and no forcing | Pushbutton operation (`Feature.Bit4` = 0): `StartAut` =1 (pos. edge) and `CancelAut` =1 | Dosing canceled |
| | Pushbutton operation (`Feature.Bit4` = 0): `ContAut` = 1 and `PauseAut` = 1 | Dosing paused |
| | Pushbutton operation (`Feature.Bit4` = 0): `StartAut` = 1 (pos. edge) and `PauseAut` = 1 | Dosing paused |
| "Manual mode" and no forcing | `StartMan` = 1 and `CancelMan` = 1 | Dosing canceled |
| | `ContMan` = 1 and `PauseMan` = 1 | Dosing paused |

### See also

Description of DoseL (Page 1305)

DoseL modes (Page 1310)

DoseL functions (Page 1312)

DoseL messaging (Page 1329)

DoseL I/Os (Page 1332)

DoseL block diagram (Page 1348)

## 7.1.5 DoseL messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Process messages
- Instance-specific messages

### Process control fault

The following control system fault messages can be output:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId2` | SIG 1 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter `CSF`. If it changes to `CSF` = 1, a control system fault is triggered (`MsgEvId02`, SIG 1).

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 1 | Alarm - high | $$BlockComment$$<br>Overdosage |
| | SIG 2 | Alarm - low | $$BlockComment$$<br>Underdosage |
| | SIG 3 | Alarm - high | $$BlockComment$$<br>PV - High alarm limit for coarse flow violated |
| | SIG4 | Alarm - low | $$BlockComment$$<br>PV - Low alarm limit for coarse flow violated |
| | SIG 5 | Alarm - high | $$BlockComment$$<br>PV - High alarm limit for coarse flow violated |
| | SIG 6 | Alarm - low | $$BlockComment$$<br>PV - Low alarm limit for fine flow violated |
| | SIG 7 | Alarm - high | $$BlockComment$$<br>PV – Creep flow too great |
| | SIG 8 | Process message - with acknowledgment | $$BlockComment$$<br>Dosing canceled |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Instance-specific messages

You can use up to three instance-specific messages with this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId2 | SIG 2 | AS process control message - fault | $$BlockComment$$<br>External message 1 |
| | SIG 3 | AS process control message - fault | $$BlockComment$$<br>External message 2 |
| | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External message 3 |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance `MsgEvId1`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | `DQ_OUT` |
| 5 | `PV_OUT` |
| 6 | `ExtVal06` |
| 7 | `ExtVa107` |
| 8 | `ExtVa108` |
| 9 | Reserved |
| 10 | Reserved |

The associated values 6 ... 8 are allocated to the parameters `ExtVa106` ... `ExtVa108` and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### Associated values for message instance `MsgEvId2`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | `ExtVa204` |
| 5 | `ExtVa205` |
| 6 | `ExtVa206` |
| 7 | `ExtVa207` |
| 8 | `ExtVa208` |
| 9 | Reserved |
| 10 | Reserved |

The associated values 4 ... 8 are allocated to the parameters `ExtVa204` ... `ExtVa208` and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of DoseL (Page 1305)

DoseL functions (Page 1312)

DoseL I/Os (Page 1332)

DoseL modes (Page 1310)

DoseL error handling (Page 1327)

DoseL block diagram (Page 1348)

Time stamp (Page 231)

## 7.1.6 DoseL I/Os

I/Os of DoseL

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutModLi* | 1= "Automatic mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutModOp* | 1 = "Automatic mode" via operator (controlled by ModLiOp = 0) | BOOL | 0 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkTypeData* | This input can not be edit or linked and is for internal usage only! | | |
| BypProt | 1 = Bypassing interlock in "local mode" and in "simulation" | BOOL | 0 |
| CancelAut* | 1 = Select Cancel in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CancelForce | 1 = Force cancel | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| CancelLocal | 1 = Select Cancel in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CancelMan* | 1 = Select Cancel in "manual mode" | BOOL | 0 |
| ContAut* | 1 = Select Continue in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ContForce | 1 = Force continue | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ContLocal | 1 = Select Continue in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ContMan* | 1 = Select Continue in "manual mode" | BOOL | 0 |
| CR_A_DC* | Creep flow: Delay time for incoming alarms [s] | REAL | 3.0 |
| CR_A_DG* | Creep flow: Delay time for outgoing alarms [s] | REAL | 3.0 |
| CR_AH_En | Creep flow:<br>1 = Enable high alarm | BOOL | 1 |
| CR_AH_Lim | Creep flow: Limit high alarm | REAL | 0.0 |
| CR_AH_MsgEn | Creep flow:<br>1 = Enable high alarm message | BOOL | 1 |
| CR_Hyst* | Creep flow: Hysteresis for alarm limit | REAL | 0.0 |
| CSF | 1 = External error (control system fault) Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CtrlChnST | Signal status of output channel Ctrl<br>Should be connected to an output channel block. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| Ctrl2ChnST | Signal status of output channel Ctrl2<br>Should be connected to an output channel block. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| DCF | Dribbling correction [%] | REAL | 25.0 |
| DosCancelMsgEn | 1 = Cancel dosing message | BOOL | 1 |
| DQ_A_DC* | Overdosage/underdosage: Delay time for incoming alarms [s] | REAL | 0.0 |
| DQ_A_DG* | Overdosage/underdosage: Delay time for outgoing alarms [s] | REAL | 0.0 |
| DQ_AH_En | Overdosage:<br>1 = Enable high alarm | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| DQ_AH_MsgEn | Overdosage:<br>1 = Enable high alarm message | BOOL | 1 |
| DQ_AH_Tol | Overdosage: Limit high alarm (relative to dosing setpoint) | REAL | 0.0 |
| DQ_AL_En | Underdosage:<br>1 = Enable low alarm | BOOL | 1 |
| DQ_AL_MsgEn | Underdosage:<br>1 = Enable low alarm message | BOOL | 1 |
| DQ_AL_Tol* | Underdosage: Limit low alarm (relative to dosing setpoint) | REAL | 0.0 |
| DQ_Ext | Dosing quantity: External setpoint | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| DQ_HiLim | Dosing quantity: High setpoint limitation | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| DQ_Int* | Dosing quantity: Internal setpoint | REAL | 0.0 |
| DQ_LoLim | Dosing quantity: Low setpoint limitation | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| DQ_OpScale | Dosing quantity: Limit for scale in bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| DQ_Unit | Unit of measure for dosing quantity | INT | 1088 |
| DQ2_Ext | Dosing quantity: External setpoint factor for fine dosing | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| DQ2_HiLim* | Dosing quantity: High limitation of setpoint factor for fine dosing | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| DQ2_Int* | Dosing quantity: Internal setpoint factor for fine dosing | REAL | 0.0 |
| DQ2_LoLim* | Dosing quantity: Low limitation of setpoint factor for fine dosing | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| DribbIn | Dribbling quantity | REAL | 0.0 |
| DribbMax | Maximum value of automatic determination of dribbling quantity | REAL | 100.0 |
| DribbCor | 1 = Automatic determination of dribbling quantity | BOOL | 0 |
| EN | 1 = Called block will be processed | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| EventTsIn | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg3 | Binary input for freely selectable message 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| ExtVa108 | Associated value 8 for messages (MsgEvID1) | ANY | |
| ExtVa204 | Associated value 4 for messages (MsgEvID2) | ANY | |
| ExtVa205 | Associated value 5 for messages (MsgEvID2) | ANY | |
| ExtVa206 | Associated value 6 for messages (MsgEvID2) | ANY | |
| ExtVa207 | Associated value 7 for messages (MsgEvID2) | ANY | |
| ExtVa208 | Associated value 8 for messages (MsgEvID2) | ANY | |
| FaultExt | 1 = External error<br>Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Feature | I/O for additional functions (Page 1312) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Feature2 | I/O for additional functions (Page 1312) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| Gain | Gain factor | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |
| Intlock | 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared<br>1 = Interlock not activated | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| Intl_En | 1 = Interlock without reset (interlock, Intlock parameter) is active | BOOL | 1 |
| LocalLi | 1 = Activate "local mode" via plant signals | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalOp* | 1 = "Local mode" via operator | BOOL | 0 |
| LocalSetting | Properties for the Local mode (Page 86) | INT | 0 |
| ManModLi* | 1 = "Manual mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManModOp* | 1 = "Manual mode" via OS operator (controlled by ModLiOp = 0) | BOOL | 1 |
| MeterType | Dosing using scales: 0=Up<br>1=Down | BOOL | 0 |
| ModLiOp | Switchover of operating mode between:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_RelOp* | 1= Release for maintenance via OS operator | BOOL | 0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgEvId2 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NumSample | Number of values for average calculation | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| U_AckLi | 1 = Acknowledgement of overdosage or underdosage via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| U_AckOp* | 1 = Acknowledgement of overdosage or underdosage via operator | BOOL | 0 |
| Occupied | 1 = In use by a batch | BOOL | 0 |
| OldDribbOut | Previous Dribbout Value | REAL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 1312) | STRUCT<br>• Bit 0: BOOL<br>• Bit 25: BOOL<br>• Bitt 26: BOOL<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1<br>• 1 |
| OS1Perm | I/O for operator permissions (Page 1312) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| P_DoseTime* | Duration of post dosing [s] | REAL | 0.0 |
| PauseAut* | 1 = Select Pause in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PauseForce | 1 = Force pause | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PauseLocal | 1 = Select Pause in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PauseMan* | 1 = Select Pause in "manual mode" | BOOL | 0 |
| Perm_En | 1 = Activation enable (enable, Permit parameter) is active | BOOL | 1 |
| Permit | 1 = Enable for opening / closing from neutral position<br>0 = Valve activation not enabled on OS | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Prot_En | 1 = Protective interlock (protection, Protect parameter) is active | BOOL | 1 |
| Protect | 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block<br>1 = Protective interlocking not activated | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| PulseWidth* | Pulse width of control signal [s] | REAL | 3.0 |
| PV | Process value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_A_DC* | Delay time for incoming PV alarms [s] when using flow dosing (coarse dosing phase) | REAL | 3.0 |
| PV_A_DG* | Delay time for outgoing PV alarms [s] when using flow dosing (coarse dosing phase) | REAL | 3.0 |
| PV_A2_DC* | Delay time for incoming PV alarms [s] when using flow dosing (fine dosing phase) | REAL | 3.0 |
| PV_A2_DG* | Delay time for outgoing PV alarms [s] when using flow dosing (fine dosing phase) | REAL | 3.0 |
| PV_AH_En | 1 = Enable PV alarm (high) limit when using flow dosing (coarse dosing phase) | BOOL | 1 |
| PV_AH_Lim | Limit PV alarm (high) when using flow dosing (coarse dosing phase) | REAL | 100.0 |
| PV_AH_MsgEn | 1 = Enable PV alarm (high) message when using flow dosing (coarse dosing phase) | BOOL | 1 |
| PV_AH2_En | 1 = Enable PV alarm (high) when using flow dosing (fine dosing phase) | BOOL | 1 |
| PV_AH2_Lim | Limit PV alarm (high) when using flow dosing (fine dosing phase) | REAL | 100.0 |
| PV_AH2_MsgEn | 1 = Enable PV alarm (high) message when using flow dosing (fine dosing phase) | BOOL | 1 |
| PV_AL_En | 1 = Enable PV alarm limit (low) when using flow dosing (coarse dosing phase) | BOOL | 1 |
| PV_AL_Lim | Limit PV alarm (low) when using flow dosing (coarse dosing phase) | REAL | 0.0 |
| PV_AL_MsgEn | 1 = Enable PV alarm (low) message when using flow dosing (coarse dosing phase) | BOOL | 1 |
| PV_AL2_En | 1 = Enable PV alarm (low) when using flow dosing (fine dosing phase) | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_AL2_Lim | Limit PV alarm (low) when using flow dosing (fine dosing phase) | REAL | 0.0 |
| PV_AL2_MsgEn | 1 = Enable PV alarm (low) message when using flow dosing (fine dosing phase) | BOOL | 1 |
| PV_Hyst* | Hysteresis for PV alarm limits when using flow dosing (coarse dosing phase) | REAL | 1.0 |
| PV_Hyst2 | Hysteresis for PV alarm limits when using flow dosing (fine dosing phase) | REAL | 1.0 |
| PV_OpScale | Limit for scale in PV bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| PV_Unit | Unit of measure for process value | INT | 1349 |
| RelaxTime* | Duration of dribbling phase [s] | REAL | 3.0 |
| RstDQ_Li | 1 = Reset dosing quantity via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstDQ_Op* | 1 = Reset dosing quantity via operator | BOOL | 0 |
| RstLi* | 1 = Reset interlock/flow monitoring via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | 1 = Reset interlock/flow monitoring via operator | BOOL | 0 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | - |
| SimDQ* | Dosing quantity: Value used in scale mode for SimOn = 1 | REAL | 0.0 |
| SimDQ_Li | Linkable simulation value DQ | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SimPV* | Process value used in flow mode for SimOn = 1 | REAL | 0.0 |
| SimPV_Li | Process value that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_Ext | External flow setpoint - coarse dosing | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_ExtLi | 1 = Select external setpoints via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtOp* | 1 = Select external setpoints via operator | BOOL | 0 |
| SP_HiLim | High limit for flow setpoint - coarse dosing | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| SP_Int* | Internal flow setpoint - coarse dosing | REAL | 0.0 |
| SP_IntLi | 1 = Select internal setpoints via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_IntOp* | 1 = Select internal setpoints via operator | BOOL | 0 |
| SP_LiOp | Select setpoint source (internal/external):<br>1 = Via interconnection<br>0 = Via operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_LoLim | Low limit for flow setpoint - coarse dosing | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_TrkExt | 1 = Bumpless switchover from external to internal setpoint active | BOOL | 1 |
| SP2_Ext | External flow setpoint - fine dosing | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP2_HiLim | High limit for flow setpoint - fine dosing | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| SP2_Int* | Internal flow setpoint - fine dosing | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP2_LoLim | Low limit for flow setpoint - fine dosing | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| StandStill | 1 = Scales at a standstill, dosing device infeedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| StartAut* | 1 = Select Start in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StartForce | 1 = Forced start | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StartLocal | 1 = Select Start in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StartMan* | 1 = Select Start in "manual mode" | BOOL | 0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| TI | Integral time [s] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |
| UA1unit | Unit of measure for analog auxiliary value 1 | INT | 0 |
| UA2unit | Unit of measure for analog auxiliary value 1 | INT | 0 |
| | | | |
| UserAna1 | Analog auxiliary value 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UserAna2 | Analog auxiliary value 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutAct | 1 = "Automatic mode" enabled<br>0 = "Manual mode" is enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutoST | 1 = Worst signal status of automatic commands | BYTE | 16#80 |
| CR_AH_Act | Creep flow: 1 = High alarm active.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl | Control output for coarse dosing | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl2 | Control output for fine dosing | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| DosEnd | 1 = End of dosing | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DosOff | 1 = Dosing off | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DosOn | 1 = Dosing on | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DosPause | 1 = Dosing pause | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DosRelax | 1 = Dosing dribbling | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DosStart | 1 = Dosing started | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| DQ_AH_Act | 1 = Overdosing<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | <br>• 0<br>• 16#80 |
| DQ_AH_TolOut | DQ_AH_Tol as output | REAL | 0.0 |
| DQ_AL_TolOut | DQ_AL_Tol as output | REAL | 0.0 |
| DQ_AL_Act | 1 = Underdosing<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | REAL | 0.0 |
| DQ_ExHiAct | Dosing quantity:<br>1 = High limit for external setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DQ_ExLoAct | Dosing quantity:<br>1 = Low limit for external setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DQ_ExtOut | Dosing quantity: External setpoint, corresponds to input parameter DQ_Ext | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| DQ_Out | Dosing quantity | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| DQ_SP | Dosing quantity setpoint used by block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| DQ_SP_Tol | Setpoint for determining tolerance limits | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| DQ_Tare | Tare memory when dosing using scales | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| DQ1_SP | Coarse dosing quantity setpoint used by block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| DQ2_ExHiAct | Dosing quantity:<br>1 = High limit for external setpoint factor for fine dosing has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| DQ2_ExLoAct | Dosing quantity:<br>1 = Low limit for external setpoint factor for fine dosing has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DQ2_ExtOut | Dosing quantity: External setpoint factor for fine dosing [%], corresponds to input parameter DQ2_Ext | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| DQ2_SP | Fine dosing quantity setpoint used by block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| DribbOut | Dribbling quantity | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| FaultExtOut | FaultExt as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see DoseL error handling (Page 1327) | INT | -1 |
| GrpErr | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IntlockOut | Intlock as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalAct | 1 = "Local mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LockAct | 1 = Interlock (Intlock, Permit or Protect) is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManAct | 1 = "Manual mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message acknowledgement status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MsgAckn2 | Message acknowledgement status 2 (output ACK_STATE of second ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | Alarm error 1 (output ERROR of first ALARM_8P) | BOOL | 0 |
| MsgErr2 | Alarm error 2 (output ERROR of second ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| MsgStat2 | Message status 2 (output STATUS of second ALARM_8P) | WORD | 16#0000 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Displays of OS_Perm | DWORD | 16#FFFFFFFF |
| OS1PermLog | Display of OS1Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1Perm | DWORD | 16#FFFFFFFF |
| P_Ctrl | 1 = Pulse output for coarse dosing | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Ctrl2 | 1 = Pulse output for fine dosing | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Rst | 1= Pulse output for reset<br>The parameter persists for one cycle after a reset. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PermitOut | Permit as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ProtectOut | Protect as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_AH_Act | 1 = PV high alarm enabled (coarse dosing phase).<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_AH2_Act | 1 = PV high alarm enabled (fine dosing phase).<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_AL_Act | 1 = PV low alarm enabled (coarse dosing phase).<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_AL2_Act | 1 = PV low alarm enabled (fine dosing phase).<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_OpScaleOut | PV_OpScale as output | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| PV_Out | Output for process value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_UnitOut | PV_Unit as output | INT | 0 |
| RdyToReset | 1 = Ready for reset via RstLi input or commands in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToStart | 1 = Active start readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP | Active flow setpoint | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_ExHiAct | Flow for coarse dosing:<br>1 = High limit for external setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExLoAct | Flow for coarse dosing:<br>1 = Low limit for external setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtAct | 1 = External setpoint is active<br>0 = Internal setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtOut | Flow for coarse dosing: External setpoint, corresponds to input parameter SP_Ext | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP1 | Coarse dosing flow setpoint used by block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP2 | Fine dosing flow setpoint used by block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP2_ExHiAct | Flow for fine dosing:<br>1 = High limit for external setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP2_ExLoAct | Flow for fine dosing:<br>1 = Low limit for external setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP2_ExtOut | Flow for fine dosing: External setpoint, corresponds to input parameter SP2_Ext | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 1305) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1305) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1305) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 1305) | DWORD | 16#00000000 |
| UA1unitOut | UA1 as output | INT | 0 |
| UA2unitOut | UA2 as output | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| `UserAna1Out` | `UserAna1` as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| `UserAna2Out` | `UserAna2` as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |

**See also**

DoseL messaging (Page 1329)
DoseL modes (Page 1310)
DoseL block diagram (Page 1348)
Error handling (Page 128)

## 7.1.7 DoseL block diagram

**DoseL block diagram**

A block diagram is not provided for this block.

**See also**

Description of DoseL (Page 1305)
DoseL modes (Page 1310)
DoseL functions (Page 1312)
DoseL error handling (Page 1327)
DoseL messaging (Page 1329)
DoseL I/Os (Page 1332)

## 7.1.8 Operator control and monitoring

### 7.1.8.1 DoseL views

#### Views of the DoseL block

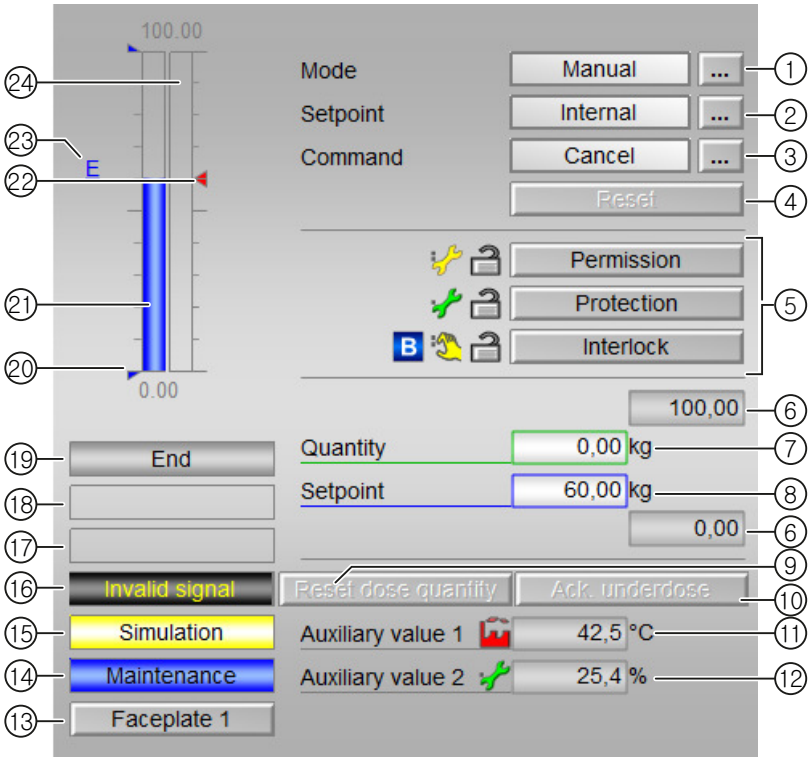
The block DoseL provides the following views:

- DoseL standard view (Page 1350)
- Alarm view (Page 336)
- DoseL limit view (Page 1354)
- Trend view (Page 341)
- DoseL parameter view (Page 1356)
- DoseL flow setpoint view (Page 1358)
- DoseL quantity setpoint view (Page 1361)
- DoseL preview (Page 1363)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for DoseL (Page 1366)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 7.1.8.2 DoseL standard view

**DoseL standard view**




**(1) Display and switch the operating mode**

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 82)
- Automatic mode (Page 82)
- Local mode (Page 86)
- Out of service (Page 77)

Refer to chapter Switching operating states and operating modes (Page 289) for information on switching the operating mode.

**(2) Displaying and changing the setpoint**

The following states can be shown and changed here:

- "Internal"
- "External"

You can find additional information on this in section Manual and automatic mode for motors, valves and dosers (Page 82).

### (3) Displaying and changing: Start, continue, pause and cancel

This area shows you the default operating state for the doser. The following states can be shown and changed here:

- "Start"
- "Continue"
- "Pause"
- "Cancel"

You can find additional information on this in section Switching operating states and operating modes (Page 289).

### (4) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in section Resetting the block in case of interlocks or errors (Page 47).

### (5) Operating range for the interlock functions of the block

You can use this button to control the interlock functions of the block. You can find additional information on this in section Interlocks (Page 107).

- Bypass information (see Forming the group status for interlock information (Page 113)):




If the text is configured for the interlocks, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (6) High and low scale range for the setpoint

This area is already set and cannot be changed.

### (7) Displaying and changing the quantity

You can find additional information on this in section Changing values (Page 291).

### (8) Displaying and changing the setpoint

You can find additional information on this in section Changing values (Page 291).

### (9) Button: Reset dose quantity

The dosing quantity can only be reset in the "End" or "Ack Dos End" state.

With resetting dose quantity also the display "Ack Dos End" is resetted to "End".

### (10) Button: Acknowledge overdosing and underdosing

Acknowledgment of overdosing and underdosing can only be made in the "Off" state.

Additional with `Feature2.Bit24` = 1:

- an acknowledgement of overdosing is possible.
- generally, overdosing or underdosing has to be acknowledged.

### (11) and (12) Display of auxiliary values

You can use this area to display two auxiliary values that have been configured in the Engineering System (ES). You can find additional information on this in section Opening additional faceplates (Page 234).

### (13) Button for switching to the standard view of any faceplate

Use this button for the standard view of a block configured in the Engineering System (ES). The visibility of this button depends on the configuration in the engineering system (ES).

You can find additional information on this in section Opening additional faceplates (Page 234).

### (14) and (15) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"
- "Maintenance"

You can find additional information on this in section Simulating signals (Page 64).

### (16) Display area for block states

This area provides additional information on the operating state of the block:

- "Internal error"
- "Changeover error"
- "Flow"
- "Invalid Signal"

### (17) Display area for block states

This area provides additional information on the operating state of the block:

- "Forced start"
- "Force continue"
- "Force pause"

- "Forced stop"
- "Request 0/1": Reset is expected but reset is not possible from faceplate. For more details, refer: Reset of errors during Request 0/1 (Page 382)

### (18) Display area for block states

This area provides additional information on the operating state of the block:

- "Underdosed"
- "Overdosed"

### (19) Display area for block states

This area provides additional information on the operating state of the block:

- "Coarse dosing"
- "Fine dosing"
- "Relax"
- "Pause"
- "Off"
- "End"
  If an underdosage or overdosage is identified, the status display shows the state "Ack Dos End" in the standard view.
- "Taring"

### (20) Limit display for the setpoint

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the ES.

### (21) Bar graph for the setpoint

This area shows the current setpoint in the form of a bar graph. You can also see the high low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (22) Limit display

These colored triangles indicate the configured limits in the respective bar graph:

- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

### (23) Display of external setpoint

This display [E] is only visible when you have selected "Internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

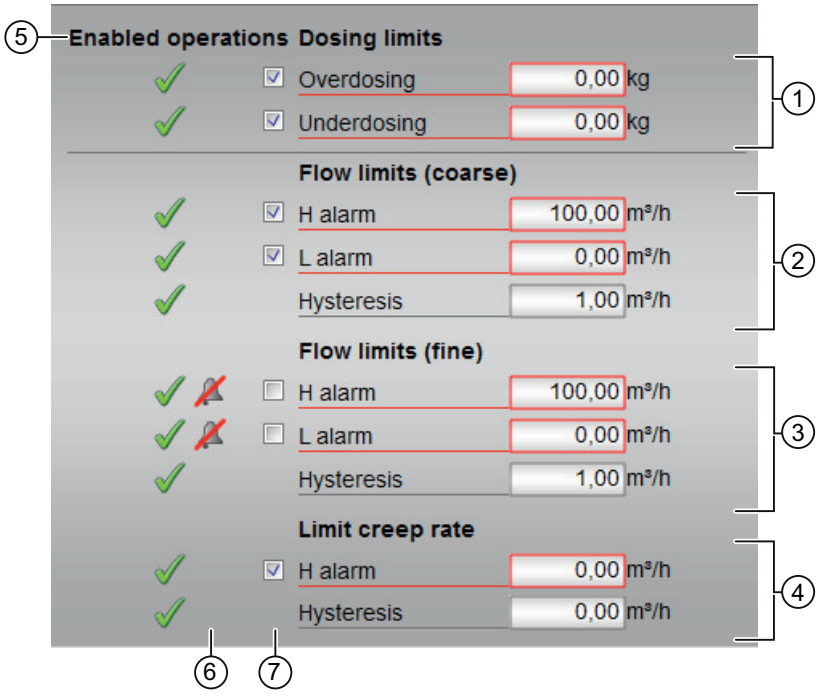
### (24) Bar graph for the quantity

This area shows you the current quantity in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

For more information referBar graph (Page 376) section.

## 7.1.8.3 DoseL limit view

### Limit view of DoseL




**Note**

In scale mode with `Feature.Bit7` = 0, the displays **(2)**, **(3)** and **(4)** are hidden.

### (1) Displaying and changing the dosing limits

You can change the dosing limits in this area:

- "Overdosing"
- "Underdosing"

You can find additional information on this in the section Changing values (Page 291).

### (2) Displaying and changing the flow limits (coarse)

You can change the flow limits (coarse) in this area:

- "H alarm"
- "L alarm"
- "Hysteresis"

You can find additional information on this in the section Changing values (Page 291).

### (3) Displaying and changing the flow limits (fine)

You can change the flow limits (fine) in this area:

- "H alarm"
- "L alarm"
- "Hysteresis"

You can find additional information on this in the section Changing values (Page 291).

### (4) Displaying and changing the limit for the creep flow

You can change the limit for the creep rate in this area:

- "H alarm"
- "Hysteresis"

You can find additional information on this in the section Changing values (Page 291).

### (5) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

### (6) "Message suppression/delay"

Message suppression indicates whether or not the suppression of the associated message in the AS block is activated with the `xx_MsgEn` parameter. The output of messages is not suppressed when the block is installed (all `xx_MsgEn` parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

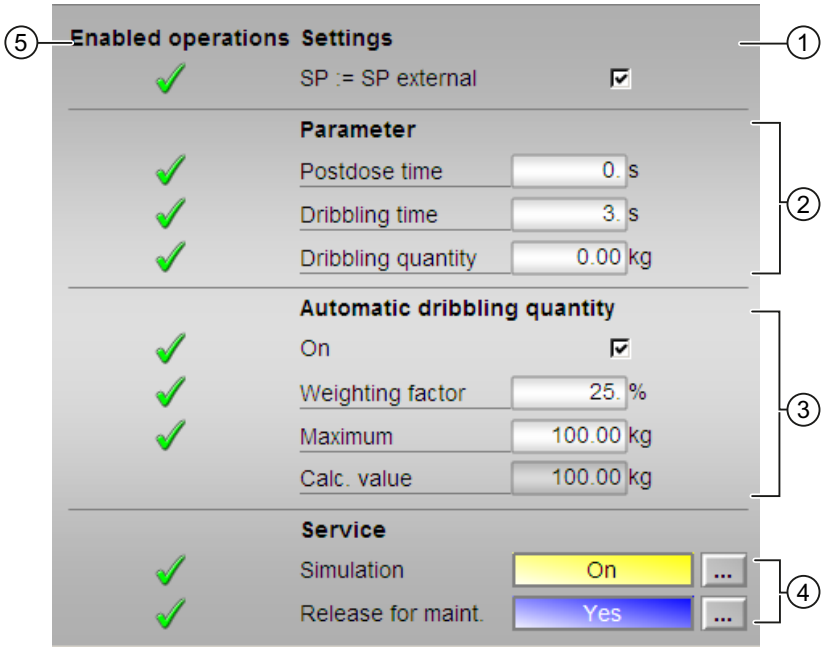
Alarm delays are also displayed in this position; for more on this see section Area of application of the alarm delays (Page 225).

### (7) Suppress messages

You can enable / disable messages by setting the check mark.

## 7.1.8.4 DoseL parameter view

### Parameter view of DoseL




### (1) Settings

You can select the following functions in this area:

- "SP:=SP external": ☑ Bumpless switchover of setpoint from external to internal. The internal setpoint is tracked to the external one.

### (2) Parameter

In this area, you change parameters and therefore influence the doser. Refer also to the Changing values (Page 291) section for more on this.

You can influence the following parameters:

- "Relax time"
- "Dribble time"
- "Dribble value"

### (3) Automatic dribbling quantity

In this area, you change the automatic dribbling quantity parameters, which affects the doser. Refer also to the Changing values (Page 291) section for more on this.

You can change the parameters when the "On" check box is selected ☑.

You can influence the following parameters:

- "Weighting factor"
- "Maximum"
- "Calc. value"

### (4) Service

You can select the following functions in this area:

- "Simulation"
- "Release for maintenance"

Refer to the Switching operating states and operating modes (Page 289) section for more on this.

You can find information on this area in the section:

- Simulating signals (Page 64)
- Release for maintenance (Page 70)

### (5) Enabled operations

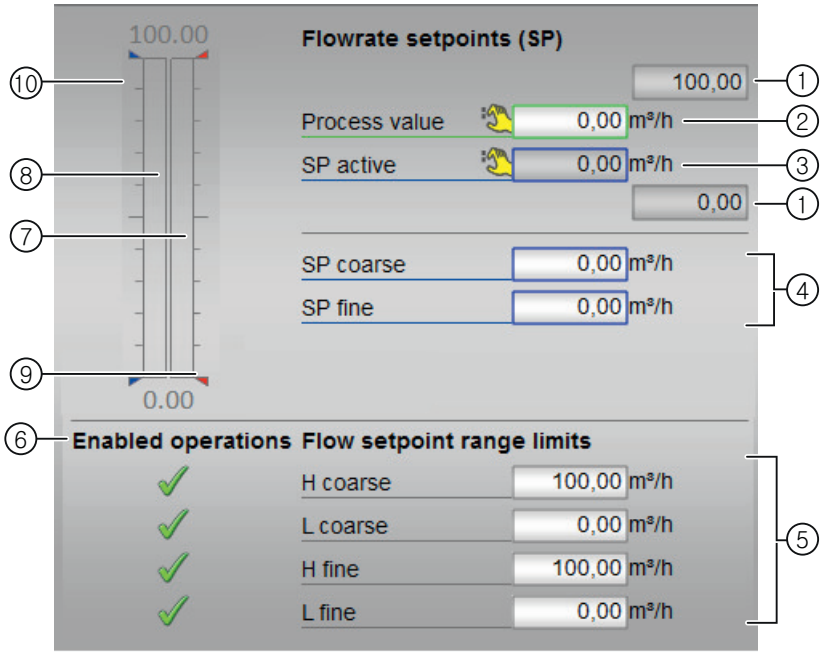
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

### 7.1.8.5 DoseL flow setpoint view

#### Flow setpoint view for DoseL




**Note**

1.) This view cannot be selected in scale mode with `Feature.Bit7` = 0 .

2.) In scale mode with `Feature.Bit7` = 1 the displays **(3)**, **(4)**, **(5)**, **(6)** and **(8)** are hidden.

3.) If `Feature.Bit15` = 1, the display of this setpoint view changes. For further information, refer to the description below.

#### (1) High and low scale range for the process value

These values provide information on the display range for the bar graph **(7)** of the process value. The scale range is defined in the engineering system.

#### (2) Display of the process value including signal status

This area shows the current process value with the corresponding signal status.

#### (3) Displaying and changing the SP active value

You can find additional information on this in the Changing values (Page 291) section.

### (4) Additional setpoint parameters

You can change the following setpoint parameters in this area:

- "SP coarse"
- "SP fine"

You can find additional information on this in the Changing values (Page 291) section.

### (5) Displaying and changing the limits

You can change the limits in this area:

- "H coarse"
- "L coarse"
- "H fine"
- "L fine"

You can find additional information on this in the Changing values (Page 291) section.

### (6) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

### (7) Bar graph for the process value

This area shows the current process value in the form of a bar graph. You can also see the high low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (8) Bar graph for the SP active setpoint

This area shows the current setpoint "SP active" in the form of a bar graph. You can also see the high low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

**Note**

If `Feature.Bit15` = 1, there are no **(8)** or **(10)** displays. For further information, refer to the description below.

### (9) Limit display

These triangles show limits for the flow limits. The flow limits are set in the limit value view.

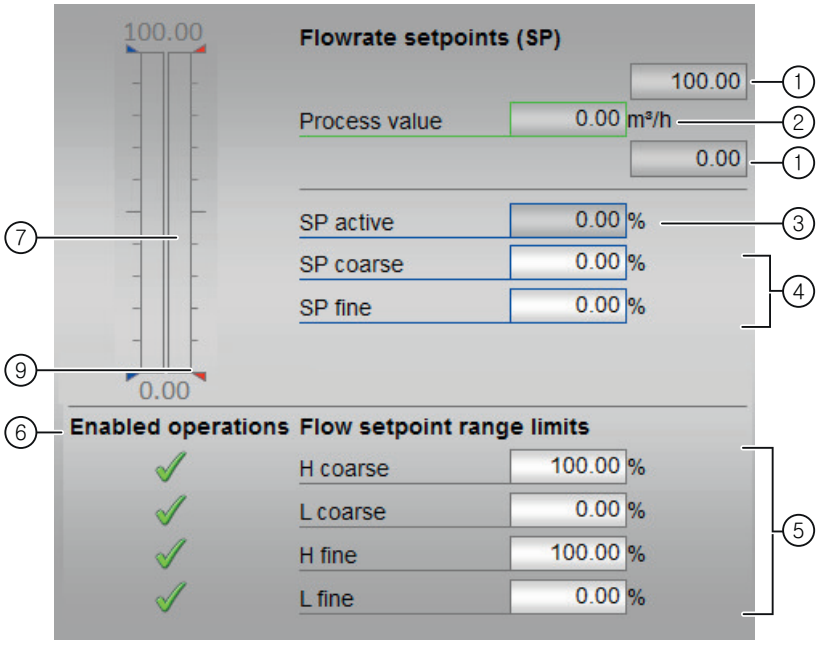
### (10) Limit display for the setpoint

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the Engineering System (ES).

### Flow setpoint view for `Feature.Bit15` = 1

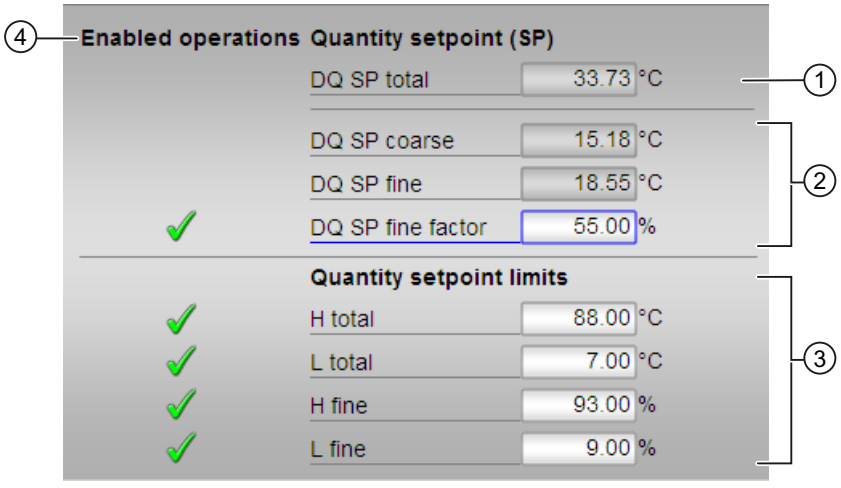
If `Feature.Bit15` = 1 , the setpoint view changes in the following ways:

- Displays **(8)** and **(10)** are omitted.
- The "SP active" value **(3)** is displayed above the area **(4)**.
- The values in the **(3)**, **(4)** and **(5)** areas contain the unit "%".



### 7.1.8.6 DoseL quantity setpoint view

#### Quantity setpoint view for DoseL




#### (1) Displaying and changing the DQ SP total setpoint

You can change the "DQ SP total" setpoint in this area.

Additional information is available in the section Changing values (Page 291).

#### (2) Displaying and changing additional setpoints

You can change the limits in this area:

- "DQ SP coarse"
- "DQ SP fine"
- "DQ SP fine factor"

Additional information is available in the section Changing values (Page 291).

**Note**

**Special note for `Feature.Bit8` = 1**

If you set this `Feature` Bit to 1, DQ SP fine is no longer displayed in the faceplate. In addition, the display of the unit for DQ SP fine factor is omitted. The input is now performed in an absolute manner.

#### (3) Displaying and changing the limits

You can change the limits in this area:

- "H total"
- "L total"

- "H fine"
- "L fine"

Additional information is available in the section Changing values (Page 291).

**Note**

**Special note for `Feature.Bit8` = 1**

If you set this `Feature` Bit to 1, the display of the unit for H fine and L fine is switched to the unit configured at `DQ_Unit`. The input is now performed in an absolute manner.

### (4) Enabled operations

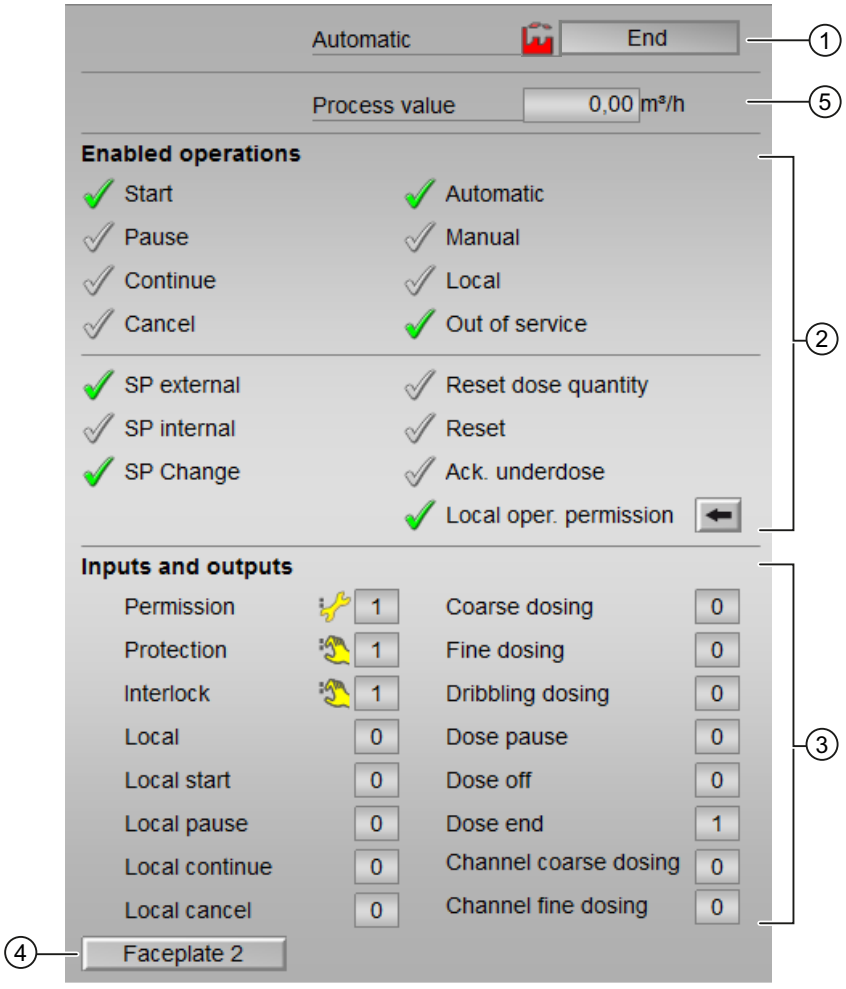
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: The OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

### 7.1.8.7 DoseL preview

#### Preview of DoseL




#### (1) Automatic preview

This area shows you the block status after it has switched from "manual" to "automatic" mode.

If the block is in "automatic mode", the current block state is displayed.

**Note**

- If the valve is closed and interlock locking is active (`SafePos`=0 Close), the automatic preview does not show the current status "Closed", instead it shows the active automatic command "Open".
- If the valve is in the end position and the automatic stop command is active, the automatic preview does not show the current status "Closed", but the active automatic command "Stop".

This area also displays the worst signal status of the following automatic commands:

- `StartAut`
- `CancelAut`
- `PauseAut`
- `ContAut`
- `SP_Ext`

### (2) Enable operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

The following enabled operations are shown here:

- "Start": You can start the dosing procedure
- "Pause": You can pause the dosing procedure
- "Continue": You can continue the dosing procedure after a pause or an abort.
- "Cancel": You can abort the dosing procedure
- "Automatic": You can switch to "automatic mode".
- "Manual": You can switch to "manual mode".
- "Local": You can switch to "local mode".
- "Out of service": You can switch to "Out of service" operating mode.
- "SP external": You can use the external setpoint
- "SP internal": You can use the internal setpoint
- "Change SP": You can change the setpoint
- "Reset dose quantity": You can reset the dosing quantity
- "Reset": You can reset the block after interlocks or errors
- "Acknowledge underdosing": You can acknowledge underdosing
- "Local operator permission": Use the ← button to switch to the standard view of the OpStations block. You can find additional information on this in the section Operator control permissions (Page 285).

### (3) Displaying current control signals

This area shows the most important parameters for this block with the current selection:

- "Permission":
  - 0 = No OS release for energizing motor
  - 1 = Enable for "starting"/"stopping" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection":
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock":
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Local": 1 = Control signal for "Local mode" (LocalLi) is active
- "Local start": 1 = Block is started in "local mode"
- "Local pause": 1 = Dosing paused in "local mode"
- "Local continue": 1 = Dosing is continued in "local mode"
- "Local cancel": 1 = Dosing is canceled in "local mode"
- "Coarse dosing": 1 = Coarse dosing is performed
- "Fine dosing": 1 = Fine dosing is performed
- "Relax phase": 1 = Dosing procedure is in the relax phase
- "Dose pause": 1 = Dosing pause
- "Dose off": 1 = No dosing taking place
- "Dose end": 1 = Dosing is stopped
- "Channel Coarse Dosing": Signal from the output channel block for "Coarse dosing"
- "Channel Fine Dosing": Signal from the output channel block for "Fine dosing"

### (4) Navigation button for switching to the standard view of any faceplate

Use this navigation button to open the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234).

### (5) Process value

This area displays the real process value (PV).

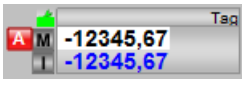
## 7.1.8.8 Block icon for DoseL

### Block symbols for DoseL

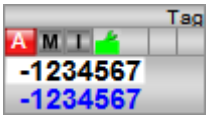
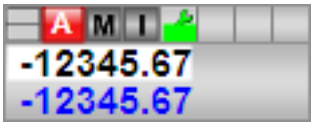
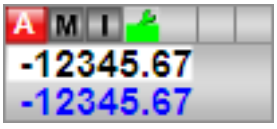
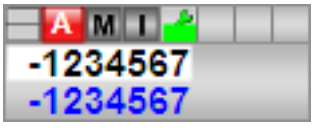
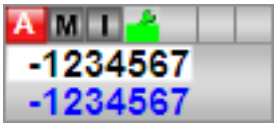
A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as control system faults
- Operating modes
- Internal and external setpoint specification
- Signal status, release for maintenance
- Track, force, and bypass
- Interlocks
- Memo display
- Process value (black, with and without decimal places)
- Setpoint (blue, with and without decimal places)

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | Block icon in the full display |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | |
|  | 6 | |
|  | 7 | |
|  | 8 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269).

# Motor and valve blocks

# 8

## 8.1 Comparison of large & small blocks

### 8.1.1 MotL compared to MotS

#### Comparison of the MotL and MotS blocks

The following tables are intended to help you decide which block to use.

#### Memory and runtime savings of the small block compared to the large block

You save the following resources for each instance:

- Memory space: ~ 40%
- Runtime: ~ 25%

#### Block operating modes

| | MotL | MotS |
|---|---|---|
| Local mode (Page 86) | X | X |
| Manual mode (Page 82) | X | X |
| Automatic mode (Page 82) | X | X |
| Out of service (Page 77) | X | X |

#### Functions of the blocks

| | MotL | MotS |
|---|---|---|
| Opening additional faceplates (Page 234) | X | X |
| Operator control permissions (Page 285) | X | X |
| Limit monitoring of an additional analog value (Page 98) | X | |
| Limit monitoring with hysteresis (Page 104) | X | |
| Suppressing messages using the MsgLock parameter (Page 231) | X | X |
| Interlocks (Page 107) | X | X |
| Motor protection function (Page 106) | X | X |

| | MotL | MotS |
|---|---|---|
| Disabling interlocks (Page 111) | X | X |
| Rapid stop for motors (Page 115) | X | |
| Resetting the block in case of interlocks or errors (Page 47) | X | X |
| Outputting group errors (Page 130) | X | X |
| Outputting a signal for start readiness (Page 59) | X | X |
| Restart lock after switching off the motor (Page 1382) | X | |
| Forming the group status for interlock information (Page 113) | X | X |
| Forming and outputting the signal status for technologic blocks (Page 117) | X | X |
| Forcing operating modes (Page 45) | X | |
| Monitoring the feedbacks (Page 104) | X | X |
| Release for maintenance (Page 70) | X | X |
| Specifying warning times for control functions at motors and valves (Page 56) | X | |
| Simulating signals (Page 64) | X | X |
| Selecting a unit of measure (Page 240) | X | |
| Neutral position for motors, valves and controllers (Page 54) | X | X |
| Output signal as a static signal or pulse signal (Page 57) | X | X |
| Generating instance-specific messages (Page 230) | X | X |
| Displaying auxiliary values (Page 239) | X | |
| Time stamp | X | |
| SIMATIC BATCH functionality (Page 73) | X | |
| Labeling of buttons and text (Page 235) | X | X |
| Displaying current control signals from output channel block | X | X |
| Display active monitoring time | X | X |
| Control priority in the event of invalid or not plausible input commands (Page 186) | X | X |
| Separate monitoring time for stopping the motor (Page 180) | X | X |

## Configurable functions using the `Feature` parameter

| Bit number | `Feature` bit function | MotL | MotS |
|---|---|---|---|
| 0 | Set startup characteristics (Page 148) | X | X |
| 1 | Reaction to the out of service mode (Page 188) | X | X |
| 2 | Resetting the commands for changing the mode (Page 172) | X | X |
| 3 | Enabling resetting of commands for the control settings (Page 172) | X | X |
| 4 | Setting switch or button mode (Page 177) | X | |
| 9 | Resetting via input signals in the event of interlocking (Protection) or errors (Page 174) | X | X |
| 10 | Exiting local mode (Page 189) | X | X |
| 11 | Activating the run time of feedback signals (Page 164) | X | X |
| 13 | Separate monitoring time for stopping the motor (Page 180) | X | |
| 14 | Enabling rapid stop via faceplate (Page 179) | X | |
| 17 | Enabling bumpless switchover to automatic mode for valves, motors, and dosers (Page 184) | X | |
| 19 | Reset even with locked state (Page 176) | X | X |
| 21 | Enable bumpless switchover to "Automatic" mode for operator only (Page 183) | X | |
| 22 | Update acknowledgment and error status of the message call (Page 171) | X | |
| 24 | Enabling local operator authorization (Page 169) | X | X |
| 25 | Suppression of all messages (Page 185) | X | X |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) | X | |
| 27 | Interlock display with LocalSetting 2 or 4 (Page 189) | X | X |
| 28 | Disabling operating points (Page 156) | X | |
| 29 | Signaling limit violation (Page 181) | X | |
| 30 | Set reset depending on the operating mode or the LiOp parameter (Page 174) | X | X |
| 31 | Activating reset of protection / error in manual mode (Page 175) | X | X |

### Configurable functions using the `Feature2` parameter

| Bit number | `Feature2` bit function | MotL | MotS |
|---|---|---|---|
| 3 | Control priority in the event of invalid or not plausible input commands (Page 186) | X | X |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) | X | X |

## 8.1.2 VlvL compared to VlvS

### Comparison of the VlvL and VlvS blocks

The following tables are intended to help you decide which block to use.

### Memory and runtime savings of the small block compared to the large block

You save the following resources for each instance:

- Memory space: ~ 25%
- Runtime: ~ 20%

### Modes

| | VlvL | VlvS |
|---|---|---|
| Local mode (Page 86) | X | X |
| Manual mode (Page 82) | X | X |
| Automatic mode (Page 82) | X | X |
| Out of service (Page 77) | X | X |

### Functions

| | VlvL | VlvS |
|---|---|---|
| Opening additional faceplates (Page 234) | X | X |
| Operator control permissions (Page 285) | X | X |
| Interlocks (Page 107) | X | X |
| Disabling interlocks (Page 111) | X | X |
| Resetting the block in case of interlocks or errors (Page 47) | X | X |

| | VlvL | VlvS |
|---|---|---|
| Outputting group errors (Page 130) | X | X |
| Outputting a signal for start readiness (Page 59) | X | X |
| Forming the group status for interlock information (Page 113) | X | X |
| Forming and outputting the signal status for technologic blocks (Page 117) | X | X |
| Forcing operating modes (Page 45) | X | |
| Monitoring the feedbacks (Page 104) | X | X |
| Disabling feedback for valves (Page 106) | X | |
| Suppressing messages using the MsgLock parameter (Page 231) | X | X |
| Release for maintenance (Page 70) | X | X |
| Specifying warning times for control functions at motors and valves (Page 56) | X | |
| Simulating signals (Page 64) | X | X |
| Selecting a unit of measure (Page 240) | X | |
| Neutral position for motors, valves and controllers (Page 54) | X | X |
| Generating instance-specific messages (Page 230) | X | X |
| Displaying auxiliary values (Page 239) | X | |
| SIMATIC BATCH functionality (Page 73) | X | X |
| Output signal as a static signal or pulse signal (Page 57) | X | |
| Time stamp (Page 2046) | X | |
| Labeling of buttons and text (Page 235) | X | X |
| Displaying current control signals from output channel block | X | X |

## Configurable functions using the `Feature` parameter

| Bit number | `Feature` bit function | VlvL | VlvS |
|---|---|---|---|
| 0 | Set startup characteristics (Page 148) | X | X |
| 1 | Reaction to the out of service mode (Page 188) | X | X |
| 2 | Resetting the commands for changing the mode (Page 172) | X | X |
| 3 | Enabling resetting of commands for the control settings (Page 172) | X | X |
| 4 | Setting switch or button mode (Page 177) | X | |
| 9 | Resetting via input signals in the event of interlocking (Protection) or errors (Page 174) | X | X |
| 10 | Exiting local mode (Page 189) | X | X |
| 11 | Activating the run time of feedback signals (Page 164) | X | X |
| 17 | Enabling bumpless switchover to automatic mode for valves, motors, and dosers (Page 184) | X | |
| 21 | Enable bumpless switchover to "Automatic" mode for operator only (Page 183) | X | |
| 22 | Update acknowledgment and error status of the message call (Page 171) | X | |
| 24 | Enabling local operator authorization (Page 169) | X | X |
| 25 | Suppression of all messages (Page 185) | X | X |
| 27 | Interlock display with LocalSetting 2 or 4 (Page 189) | X | X |
| 30 | Set reset depending on the operating mode or the LiOp parameter (Page 174) | X | X |
| 31 | Activating reset of protection / error in manual mode (Page 175) | X | X |

## Configurable functions using the `Feature2` parameter

| Bit number | `Feature2` bit function | VlvL | VlvS |
|---|---|---|---|
| 3 | Control priority in the event of invalid or not plausible input commands (Page 186) | X | X |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) | X | X |

## 8.1.3 ShrdResL compared to ShrdResS

### Comparison of the ShrdResL and ShrdResS blocks

The following tables are intended to help you decide which block to use.

### Memory and runtime savings of the small block compared to the large block

You save the following resources for each instance:

- Memory space: ~ 50%
- Runtime: ~ 25%

### Parameters for channel

| | ShrdResL | ShrdResS |
|---|---|---|
| Control commands for channel | 12 | 6 |
| External setpoints for channel | 8 | 2 |
| Additional strings for channel | 2 | - |

### Functions

| | ShrdResL | ShrdResS |
|---|---|---|
| Opening additional faceplates (Page 234) | x | x |
| Operator control permissions (Page 285) | x | |
| Readiness signal (Page 1607) | x | x |
| Channel management (Page 1629) | | x |
| Channel management with parameterized priority parameter and strategy mode (Page 1607) | x | |
| Allocate/enable channel (Page 1607) | x | x |
| Enable/disable channel (Page 1607) | x | x |
| Channel prioritization (Page 1607) | x | x |
| Cascading (Page 1629) | | x |
| SIMATIC BATCH functionality (Page 73) | x | x |

### Configurable functions using the `Feature` parameter

| Bit number | `Feature` bit function | ShrdResL | ShrdResS |
|---|---|---|---|
| 0 | Set startup characteristics (Page 148) | x | x |

# 8.2 MotL - Motor (Large)

## 8.2.1 Description of MotL

### Object name (type + number) and family

Type + number: FB 1850

Family: Drives

### Area of application for MotL

The block is used for the following applications:

- Control of motors with one control signal

**Note**

This block is also available as a small block. A comparison of the MotL and MotS blocks is available in the section: MotL compared to MotS (Page 1369)

### How it works

The block is used to control motors. Various inputs are available for controlling the motor.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section MotL I/Os (Page 1394).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |

| Status bit | Parameter |
|---|---|
| 4 | `OosLi.Value` |
| 5 | 0 = `ManAct.Value`<br>1 = `AutoAct.Value` |
| 6 | `LocalAct.Value` |
| 7 | 0: Open padlock in the block icon<br>1: Closed padlock in the block icon |
| 8 | `Start.Value` |
| 9 | `Stop.Value` |
| 10 | `MonDynStopErr.Value` |
| 11 | `MonStaErr.Value` |
| 12 | `MonDynErr.Value` |
| 13 | `BypProt` |
| 14 | Invalid signal status |
| 15 | Mode switchover error |
| 16 | 1 = `Intlock` is active |
| 17 | 1 = `Permit` is active |
| 18 | 1 = `Protect` is active |
| 19 | Trip.Value |
| 20 | Display „Forced start“ |
| 21 | Display „Forced stop“ |
| 22 | Not used |
| 23 | "Interlock" button is enabled |
| 24 | Reset request in automatic preview |
| 25 | `WarnAct.Value` or<br>`Idle Time` is active |
| 26 | Bypass information for display in block icon |
| 27 | Automatic preview for "starting" |
| 28 | Automatic preview for "stopping" |
| 29 | External error generated by FaultExt or external control system fault CSF with set `Feature.Bit18` Activating error state for external process control error CSF (Page 162) |
| 30 | "Permission" button is enabled |
| 31 | "Protection" button is enabled |

## Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `MsgLock` |
| 1 | `AV_AH_Act.Value` |
| 2 | `AV_WH_Act.Value` |
| 3 | `AV_TH_Act.Value` |
| 4 | `AV_TL_Act.Value` |
| 5 | `AV_WL_Act.Value` |

| Status bit | Parameter |
|---|---|
| 6 | AV_AL_Act.Value |
| 7 | AV_AH_En OR Feature.Bit23 |
| 8 | AV_WH_En OR Feature.Bit23 |
| 9 | AV_TH_En OR Feature.Bit23 |
| 10 | AV_TL_En OR Feature.Bit23 |
| 11 | AV_WL_En OR Feature.Bit23 |
| 12 | AV_AL_En OR Feature.Bit23 |
| 13 | AV_AH_MsgEn |
| 14 | AV_WH_MsgEn |
| 15 | AV_TH_MsgEn |
| 16 | AV_TL_MsgEn |
| 17 | AV_WL_MsgEn |
| 18 | AV_AL_MsgEn |
| 19 | 1 = No impact of input signals on "local mode" with LocalSetting = 0, LocalSetting = 2, LocalSetting = 4 and OOS mode. |
| 20 | Motor is stopped |
| 21 | Motor is stopping |
| 22 | Motor is starting |
| 23 | Motor is running |
| 24 | Error in motor |
| 25 | 1 = Input parameter StartChnST is interconnected |
| 26 | Resynchronization with state "No Cmd" |
| 27 | 1 = "Permission stop" button is enabled |
| 28 | 1 = Permission stop is active |
| 29 | Hidden bypass signal in Permission stop |
| 30 | Display for interlocks in block icon |
| 31 | MS_RelOp |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 | Delay of the AV_AH_Lim message |
| 1 | Delay of the AV_WH_Lim message |
| 2 | Delay of the AV_TH_Lim message |
| 3 | Delay of the AV_TL_Lim message |
| 4 | Delay of the AV_WL_Lim message |
| 5 | Delay of the AV_AL_Lim message |
| 6 | Collection of message delays |
| 7 - 10 | Not used |
| 11 | Hidden bypass signal in Permit |
| 12 | Hidden bypass signal in interlock |
| 13 | Hidden bypass signal in Protect |

| Status bit | Parameter |
|---|---|
| 14 | `Feature2.Bit2`: Separate bypass signal |
| 15 - 17 | Current monitoring time is visible |
| 18 | `SimLiOp.Value` |
| 19 | 1 = Enable for rapid stop (`Feature` Bit Enabling rapid stop via faceplate (Page 179)) |
| 20 - 22 | Not used |
| 23 | Command for rapid stop |
| 24 | Output command for starting the motor |
| 25 | Output command for stopping the motor |
| 26 | Show automatic preview in the standard view |
| 27 | Internal error |
| 28 | `GrpErr.Value` |
| 29 | `RdyToStart.Value` |
| 30 | Auxiliary value 1 visible |
| 31 | Auxiliary value 2 visible |

### Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 | Effective signal 1 of the message block connected via `EventTsIn` |
| 1 | Effective signal 2 of the message block connected via `EventTsIn` |
| 2 | Effective signal 3 of the message block connected via `EventTsIn` |
| 3 | Effective signal 4 of the message block connected via `EventTsIn` |
| 4 | Effective signal 5 of the message block connected via `EventTsIn` |
| 5 | Effective signal 6 of the message block connected via `EventTsIn` |
| 6 | Effective signal 7 of the message block connected via `EventTsIn` |
| 7 | Effective signal 8 of the message block connected via `EventTsIn` |
| 8 | `AV` not connected |
| 9 | Motor protection display (`Trip.Status` ≠ 16#FF) |
| 10 | 1 = Input parameter `FbkRun` is connected |
| 11 | 1 = Input parameter `FbkStop` is connected |
| 12 | Resynchronization preview for Start |
| 13 | Resynchronization preview for Stop |
| 14 | Resynchronization: Display preview |
| 15 | Resynchronization: Display small preview symbol in standard view |
| 16 - 23 | Effective signal 9..16 of the message block connected via `EventTsIn` |
| 24 - 28 | Not used |
| 29 | 1= Suppressed messages |
| 30 | ot used |
| 31 | Separate monitoring of shutdown of the motor (`Feature.Bit13`) |

### Status word allocation for `Status5` parameter

| Status bit | Parameter |
|---|---|
| 0 - 15 | Effective signal 1...16 of the message block connected via `EventTs2In` |
| 16 - 31 | Not used |

### See also

MotL functions (Page 1382)

MotL messaging (Page 1392)

MotL block diagram (Page 1404)

MotL error handling (Page 1390)

MotL modes (Page 1380)

## 8.2.2 MotL modes

### MotL operating modes

The block can be operated using the following modes:

- Local mode (Page 86)
- Automatic mode (Page 82)
- Manual mode (Page 82)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

### "Local mode"

You can find general information on "Local mode", switching modes and Bumpless switchover in the Local mode (Page 86) section.

Motor actions you can control in local mode:

- "Start" (`StartLocal` = 1)
- "Stop" (`StopLocal` = 1)

A motor operated in "local mode" is controlled either by "local" signals or by the feedback signals (the `FbkStart` = 1 or `FbkStop` =1 input parameter). Configuration takes place via the input parameter `LocalSetting`.

`LocalSetting` =2,4: The control outputs are tracked to the feedbacks.

All two feedbacks are available and plausible means that always only one feedback is active. The corresponding control output is tracked:

```
Start = FbkRun
```

```
StopCtrl = FbkStop
```

When both feedbacks are false then the drive is between of two positions. In this case no monitoring is started. When two feedbacks are active, then they are not plausible. In this case the static monitoring counter starts. In both cases "all feedbacks are reset" or "not plausible feedbacks" with `Feature2Bit21` =1 all the control outputs are set to false (No command state) else control outputs will be hold.

If one feedback is available it is always plausible.

| NoFbkRun =0 (default) | NoFbkStop =0 |
|---|---|
| `Start = FbkRun`<br>`StopCtrl = NOT FbkRun` | `Start = NOT FbkStop`<br>`StopCtrl = FbkStop` |

When all feedbacks are not available the control outputs will be hold.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and Bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 82) section.

Motor actions you can control in auto mode:

- "Start" (`StartAut` = 1)
- "Stop" (`StopAut` = 1)

### "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the section Manual and automatic mode for motors, valves and dosers (Page 82).

Motor actions you can control in manual mode:

- "Start" (`StartMan` = 1)
- "Stop" (`StopMan` = 1)

### "Out of service"

You can find general information about the "Out of service" mode in the Out of service (Page 77) section.

When the mode is changed to "Out of service", the status of the following output parameters with `Feature2.Bit1` =0 and `Feature2.Bit1` =1 is as shown below:

| Output Parameters | **`Feature2.Bit1` =0** | **`Feature2.Bit1` =1** |
|---|---|---|
| FbkRunOut.ST | 16#80 | 16#60 |
| Run.ST | 16#80 | 16#60 |
| Stop.ST | 16#80 | 16#60 |
| AV_Out.ST | 16#80 | 16#60 |

### See also

MotL block diagram (Page 1404)
MotL I/Os (Page 1394)
MotL messaging (Page 1392)
MotL error handling (Page 1390)
MotL functions (Page 1382)
Description of MotL (Page 1376)

## 8.2.3 MotL functions

### Functions of MotL

The functions for this block are listed below.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" |
| 1 | 1 = Operator can switch to "manual mode" |
| 2 | 1 = Operator can switch to "local mode" |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can stop the motor |
| 5 | 1 = Operator can start the motor |
| 6 | Not used |
| 7 | 1 = Operator can reset the motor |
| 8 | 1 = Operator can define or change the monitoring time for startup |
| 9 | 1 = Operator can define the monitoring time for the status |
| 10 | 1 = Operator can enable the monitoring time function (Bit 8 - 9) |
| 11 | 1 = Operator can activate the Simulation function |
| 12 | 1 = Operator can activate the Release for maintenance function |
| 13 | 1 = Operator can change the high limit (`AV`) for the alarm |
| 14 | 1 = Operator can change the high limit (`AV`) for the warning |
| 15 | 1 = Operator can change the high limit (`AV`) for the tolerance |
| 16 | 1 = Operator can change the limit (`AV`) for hysteresis |

| Bit | Function |
|---|---|
| 17 | 1 = Operator can lower the limit (AV) for the alarm |
| 18 | 1 = Operator can lower the limit (AV) for the warning |
| 19 | 1 = Operator can lower the limit (AV) for the tolerance |
| 20 | 1 = Operator can activate / deactivate messages via AV_AH_MsgEn |
| 21 | 1 = Operator can activate / deactivate messages via AV_WH_MsgEn |
| 22 | 1 = Operator can activate / deactivate messages via AV_TH_MsgEn |
| 23 | 1 = Operator can activate / deactivate messages via AV_TL_MsgEn |
| 24 | 1 = Operator can activate / deactivate messages via AV_WL_MsgEn |
| 25 | 1 = Operator can activate / deactivate messages via AV_AL_MsgEn |
| 26 | 1 = Operator can change the simulation value SimAV |
| 27- 29 | Not used |
| 30 | 1 = Operator can define the monitoring time for stopping |
| 31 | 1 = Operator can execute the rapid stop |

The block has the following permissions for the OS1_Perm parameter:

| Bit | Function |
|---|---|
| 0 - 15 | reserved |

**Note**

If you interconnect a parameter that is also listed in OS_Perm as a parameter, you have to reset the corresponding OS_Perm bit.

### Limit monitoring of an additional analog value

This block provides the standard function Limit monitoring of an additional analog value (Page 98).

### Limit monitoring with hysteresis

This block provides the standard function Limit monitoring with hysteresis (Page 104).

### Suppressing messages using the MsgLock parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Interlocks

This block provides the following interlocks:

- Activation enable
- Interlock without reset ("Interlock")

- Interlock with reset ("Protection")
- Permission for stop ("Permission stop")

Refer to the section Interlocks (Page 107) as well as Influence of the signal status on the interlock (Page 112).

### Motor protection function

This block provides the standard function Motor protection function (Page 106).

### Disabling interlocks

This block provides the standard function Disabling interlocks (Page 111).

### Rapid stop for motors

This block provides the standard function Rapid stop for motors (Page 115).

### Resetting the block in case of interlocks or errors

This block provides the standard function Resetting the block in case of interlocks or errors (Page 47).

### External error (`FaultExt`), external control system fault (`CSF`)

This block provides the possibility to pass an external error via the `FaultExt` parameter or an external control system fault via the CSF parameter. See MotL error handling (Page 1390)

### Group error

This block provides the standard function Outputting group errors (Page 130).

The following parameters are taken into consideration when forming the group error:

- `CSF`
- `Trip`
- `MonDynErr`
- `MonStaErr`
- `FaultExt`

### Outputting a signal for start readiness

This block provides the standard function Outputting a signal for start readiness (Page 59).

### Restart lock after switching off the motor

After switching off or stopping the motor, it can only be restarted after the time set with the `IdleTime` input parameter.

When the "Stop" command is given, the motor goes immediately into "Stop" mode, and `IdleTime`starts after the feedback (`FbkRun` = 0) is given. The motor cannot be started again until the `IdleTime` has expired.

The `IdleTime` parameter can be set independently of the `MonTiDynamic` parameter.

### Forming the group status for interlocks

This block provides the standard function Forming the group status for interlock information (Page 113).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `LocalLi.ST`
- `StartLocal.ST`
- `StopLocal.ST`
- `Trip.ST`
- `FbkRunOut.ST`
- `AV_Out.ST`
- `StartChn.ST`
- `StartAut.ST` (only if `Feature2.Bit10` = 1)
- `StopAut.ST` (only if `Feature2.Bit10` = 1)

### Considering bad quality of automatic commands or external values

If the `Feature2` bit Considering bad quality of automatic commands or external values (Page 200) is set to 1 and one of the following parameters has bad signal status (16#00 or 16#28), the block goes to the neutral position (motor stop) in the "Automatic" mode:

- `StartAut.ST`
- `StopAut.ST`

### Forcing operating modes

This block provides the standard function Forcing operating modes (Page 45).

The following states can be enforced:

- "Start" (`StartForce`)
- "Stop" (`StopForce`)

### Feedback monitoring

This block provides the standard functio

### Disabling feedback

This block provides the standard function of Disabling feedback for valve and motors (Page 106). Feedback monitoring can be deactivated separately for each feedback with `NoFbkRun` or `NoFbkStop` as required.

The output signals of feedback are tracked by the control and the other available feedbacks:

| NoFbkRun | NoFbkStop | Comment |
|---|---|---|
| 0 | 1 | **Default:**<br>`FbkRunOut = FbkRun;`<br>`FbkStopOut = StopCtrl AND NOT FbkRun;`<br>**Local mode with LocalSetting =2/4:**<br>`FbkRunOut = FbkRun;`<br>`FbkStopOut = NOT FbkRun;` |
| 0 | 0 | `FbkRunOut = FbkRun;`<br>`FbkStopOut = FbkStop;` |
| 1 | 0 | `FbkRunOut = Start AND NOT FbkStop;`<br>`FbkStopOut = FbkStop;`<br>**Local mode with LocalSetting =2/4:**<br>`FbkRunOut = NOT FbkStop;`<br>`FbkStopOut = FbkStop;` |
| 1 | 1 | `FbkRunOut = Start;`<br>`FbkStopOut = StopCtrl;` |

When all feedbacks are not available, the position is on hold after a switch to local mode with `LocalSetting` =2/4.

### Using StopCtrl with mode "No Cmd"

This block provides the standard function Control with output StopCtrl and state “No Cmd” (Page 214)

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Specify warning times for control functions

This block provides the standard function Specifying warning times for control functions at motors and valves (Page 56).

You can generate warning signals when, for example, motors are started. Warning signals can be generated in the following modes:

- Manual mode (Page 82) (`WarnTiMan` input parameter)
- Automatic mode (Page 82) (`WarnTiAut` input parameter)

You specify the warning times in seconds using the input parameters `WarnTiMan` and `WarnTiAut`. If, for example, a motor is started, then this is displayed at the output parameter with `WarnAct` = 1. The motor then starts after the set warning time has expired and `WarnAct` then goes back to 0.

A corresponding warning is not output if the warning times (`WarnTiMan` or `WarnTiAut`) are specified with a smaller value than the `SampleTime` parameter.

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Additional value (`SimAV`, `SimAV_Li`)

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

### Neutral position

This block provides the standard function Neutral position for motors, valves and controllers (Page 54).

### Output signal as a pulse signal or static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 57).

### Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 230).

### Resynchronization

This block provides the standard function Resynchronization (Page 258).

Resynchronization is switched on over feature bits:

| | |
|---|---|
| Feature2 Bit17 =1 | Resynchronization after CPU startup |
| Feature2 Bit18 =1 | Resynchronization after Simulation or after Out of service |
| Feature2 Bit19 =1 | Resynchronization after control system fault |
| Feature2 Bit20 =1 | Resynchronization with 0->1 trigger at StateTrkOn input |

If there are no restrictions the control outputs are tracked as follow:

**NoFbkRun =0, NoFbkStop =1 (default):**

Start := FbkRun;

StopCtrl := NOT FbkRun;

**NoFbkRun =0, NoFbkStop =0:**

Plausible feedback positions (FbkRun =1, FbkStop =0 or FbkRun =0, FbkStop =1) are tracked as control position.

Start := FbkRun;

StopCtrl := FbkStop;

In case there are no plausible feedback positions (FbkRun =0, FbkStop =0 or FbkRun =1, FbkStop =1) the control position depends on Feature2 Bit21.

- Feature2 Bit21 =0 (default): Control position are not changed.
- Feature2 Bit21 =1: Control position becomes "No Cmd" Start := 0; StopCtrl := 0;

**NoFbkRun =1, NoFbkStop =0:**

Start := NOT FbkStop;

StopCtrl := FbkStop;

With other combinations of **NoFbkRun**, **NoFbkStop** the controls are not changed.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the section Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 3 | Enabling resetting of commands for the control settings (Page 172) |
| 4 | Setting switch or button mode (Page 177) |
| 9 | Resetting via input signals in the event of interlocking (Protection) or errors (Page 174) |
| 10 | Exiting local mode (Page 189) |
| 11 | Activating the run time of feedback signals (Page 164) |
| 14 | Enabling rapid stop via faceplate (Page 179) |

| Bit | Function |
|---|---|
| 13 | Separate monitoring time for stopping the motor (Page 180) |
| 17 | Enabling bumpless switchover to automatic mode for valves, motors, and dosers (Page 184) |
| 18 | Activating error state for external process control error CSF (Page 162) |
| 19 | Reset even with locked state (Page 176) |
| 20 | Disable calculation of impulse control in LocalSetting 2 and 4 (Page 200) |
| 21 | Enable bumpless switchover to "Automatic" mode for operator only (Page 183) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 23 | Limits disabled in faceplate (Page 206) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 27 | Interlock display with LocalSetting 2 or 4 (Page 189) |
| 28 | Disabling operating points (Page 156) |
| 29 | Signaling limit violation (Page 181) |
| 30 | Set reset depending on the operating mode or the LiOp parameter (Page 174) |
| 31 | Activating reset of protection / error in manual mode (Page 175) |

In pushbutton operation (Bit 4 = 0) the automatic commands in "automatic" mode are latching, in other words `StartAut`, `StopAut` can be reset to 0 after changing the control. In "manual" and "local" modes, however, the automatic commands are not saved and in the absence of automatic commands the automatic control is tracked.

In switching mode (Bit 4 = 1), the control is selected with the static signal `StartAut`. If `StartAut` input is not set the motor is stopped. Control via `StopAut` is not needed. If the "Activate command reset for control" function (Bit 3 = 1) is also activated, the `StartAut` input is reset to the neutral position after evaluation in the block.

### Configurable reactions using the `Feature2` parameter

| Bit | Function |
|---|---|
| 1 | Changes signal status of outputs in OOS (Page 208) |
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 3 | Control priority in the event of invalid or not plausible input commands (Page 186) |
| 4 | Setting switch or button mode for local commands (Page 195) |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) |
| 8 | Forcing commands in the Local mode (Page 198) |
| 9 | Button mode with triggered local control commands (Page 209) |
| 10 | Considering bad quality of automatic commands or external values (Page 200) |
| 11 | Suppress MsgLock and “Out of service” mode for a connected message block (Page 202) |
| 13 | Resetting the Pending Error in Local Mode 1 and 3 (Page 210) |
| 17 | Resynchronization after startup (Page 213) |
| 18 | Resynchronization after out of service or after simulation (Page 213) |
| 19 | Resynchronization after CSF (Page 214) |
| 20 | Resynchronization with StateTrkOn input (Page 214) |
| 21 | Control with output StopCtrl and state “No Cmd” (Page 214) |

### Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 239).

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Button labels

This block provides the standard function Labeling of buttons and text (Page 235).

Instance-specific text can be configured for the following parameters:

- `StartMan`
- `StopMan`
- `RapidStp`

### See also

MotL messaging (Page 1392)

MotL I/Os (Page 1394)

MotL block diagram (Page 1404)

MotL modes (Page 1380)

Monitoring the feedbacks (Page 104)

Description of MotL (Page 1376)

## 8.2.4 MotL error handling

### Error handling of MotL

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Mode switchover error
- Control system fault (CSF)
- Invalid or not plausible input commands

### Overview of error numbers

The ErrorNum I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed. |
| 0 | There is no error. |
| 41 | The value for the LocalSetting I/O is not within the approved limit of 0 to 4. |
| 42 | LocalSetting = 0 or LocalSetting = 3 or LocalSetting = 4 and LocalLi = 1 |
| 51 | StartLocal = 1 and StopLocal = 1<br>StartAut = 1 and StopAut = 1<br>AutModLi = 1 and ManModLi = 1<br>StartForce = 1 and StopForce = 1 |
| 52 | LocalAct = 1 and LocalSetting = 2 or 4 and SimOn = 1 |

### Mode switchover error

This error can be output by the block, see the section Error handling (Page 128).

### Invalid or not plausible input commands

This error can be output by the block, see the section Error handling (Page 128).

For the following invalid or not plausible input commands, the control output can be kept or switched to the neutral position. This depends on the function Control priority in the event of invalid or not plausible input commands (Page 186).

| Operating mode | Invalid or not plausible input commands | Control reaction with Feature2.Bit3 = 1 |
|---|---|---|
| Local: Localsetting = 1 or Localsetting = 3 | Pushbutton operation for local mode (Feature2.Bit4 = 0):<br>StartLocal = 1 and StopLocal = 1 | Motor is stopped |
| Local: Localsetting = 1 or Localsetting = 3 and forcing | StartForce = 1 and StopForce = 1 | |
| Forcing and no "local mode" | StartForce = 1 and StopForce = 1 | |
| "Automatic mode" and no forcing | Pushbutton operation (Feature.Bit4 = 0):<br>StartAut = 1 and StopAut = 1 | |
| "Manual mode" and no forcing | StopMan = 1 and StartMan = 1 | |

### Control system fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 128) section for more on this.

### See also

MotL block diagram (Page 1404)
MotL I/Os (Page 1394)
MotL messaging (Page 1392)
MotL functions (Page 1382)
MotL modes (Page 1380)
Description of MotL (Page 1376)

## 8.2.5 MotL messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Instance-specific messages

### Process control fault

The following control system error messages can be output:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId1` | SIG 1 | AS process control message - fault | $$BlockComment$$<br>Motor feedback error |
| | SIG 2 | AS process control message - fault | $$BlockComment$$<br>Motor protection triggered |
| | SIG 3 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter `CSF`. If it changes to `CSF` = 1, a process control fault is triggered (`MsgEvId1`, SIG 3).

### Instance-specific messages

You can use up to three instance-specific messages with this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External message 1 |
| | SIG 5 | AS process control message - fault | $$BlockComment$$<br>External message 2 |
| | SIG 6 | AS process control message - fault | $$BlockComment$$<br>External message 3 |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance **MsgEvId1**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |
| 4 | ExtVa104 |
| 5 | ExtVa105 |
| 6 | ExtVa106 |
| 7 | ExtVa107 |
| 8 | ExtVa108 |
| 9 | Reserved |
| 10 | Reserved |

The associated values 4 ... 8 are allocated to the parameters ExtVa104 ... ExtVa108 , and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of MotL (Page 1376)

MotL functions (Page 1382)

MotL I/Os (Page 1394)

MotL block diagram (Page 1404)

MotL error handling (Page 1390)

MotL modes (Page 1380)

## 8.2.6 MotL I/Os

I/Os of MotL

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutModLi* | 1 = "Automatic mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutModOp* | 1 = "Automatic mode" via operator (controlled by ModLiOp = 0) | BOOL | 0 |
| AV | Input additional analog value, to be connected to AV_Tech of the AV block | ANY | |
| AV_AH_Lim | Limit high alarm | REAL | 95.0 |
| AV_AL_Lim | Limit low alarm | REAL | 5.0 |
| AV_Hyst | Hysteresis for alarm, warning and tolerance limits | REAL | 1.0 |
| AV_TH_Lim | Limit high tolerance | REAL | 85.0 |
| AV_TL_Lim | Limit low tolerance | REAL | 15.0 |
| AV_WH_Lim | Limit high warning | REAL | 90.0 |
| AV_WL_Lim | Limit low warning | REAL | 10.0 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkTypeData* | This input can not be edit or linked and is for internal usage only! | | |
| BypProt | 1 = Bypassing interlock in "local mode" and in "simulation" | BOOL | 0 |
| CSF | 1 = External error (control system fault) Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| EventTsIn | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| EventTs2In | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTs2In input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg3 | Binary input for freely selectable message 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa104 | Associated value 4 for messages (MsgEvID1) | ANY | |
| ExtVa105 | Associated value 5 for messages (MsgEvID1) | ANY | |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| ExtVa108 | Associated value 8 for messages (MsgEvID1) | ANY | |
| FaultExt | 1 = External error<br>Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| FbkRun | 1= Motor feedback running | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| FbkStop | 1= Motor feedback stopped | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| Feature | I/O for additional functions (Page 1382)<br><br>Separate monitoring time for stopping the motor | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 13: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0<br>• 0<br>• 0 |
| Feature2 | I/O for additional functions (Page 1382) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| IdleTime* | Wait time for restart in [s] | REAL | 5.0 |
| Intlock | 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared<br>1 = Interlock not activated | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| Intl_En | 1 = Interlock without reset (interlock, Intlock parameter) is active | BOOL | 1 |
| LocalLi | 1 = Activate "local mode" via plant signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalOp* | 1 = "Local mode" via operator | BOOL | 0 |
| LocalSetting | Properties for the Local mode (Page 86) | INT | 0 |
| ManModLi* | 1 = "Manual mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManModOp* | 1 = "Manual mode" via OS operator (controlled by ModLiOp = 0) | BOOL | 1 |
| ModLiOp | Switchover of operating mode between:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Monitor | 1 = Feedback monitoring | BOOL | 1 |
| MonSafePos | 1 = Go to neutral position in the event of monitoring errors  Motors: Stop is the neutral position | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MonTiDynamic* | Monitoring time for feedback errors or feedback start error after successful operation in [s] | REAL | 3.0 |
| MonTiDyStop* | Monitoring time for feedback stop errors after successful operation in [s] | REAL | 3.0 |
| MonTiStatic* | Monitoring time for feedback errors without operation in [s] | REAL | 3.0 |
| MS_RelOp* | 1= Release for maintenance via OS operator | BOOL | 0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NoFbkRun | 1= Motor feedback running not present | BOOL | 0 |
| NoFbkStop | 1= Motor feedback stopped not present | BOOL | 1 |
| Occupied | 1 = In use by a batch | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 1382)<br>1 = Operator can set or change the monitoring time for "Control: Start"<br>1 = Operator can set or change the monitoring time for "Control: Stop" | STRUCT<br>• Bit 0: BOOL<br>• Bit 8: BOOL<br>• Bit 20: BOOL<br>• Bit 30: BOOL | -<br>• 1<br>• 1<br>• 1<br>• 1 |
| OS1Perm | I/O for operator permissions (Page 1382) | STRUCT<br>• Bit 0: BOOL<br>• Bit 15: BOOL | -<br>• 1<br>• 1 |
| Permit | 1 = OS activation enable for motor<br>0 = No OS release for energizing motor | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PermitSafePos | 1=Motor stop is allowed | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL<br>(Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| PermSf_En | 1=Permit stop enabled | BOOL | 1 |
| Perm_En | 1 = Activation enable (enable, Permit parameter) is active | BOOL | 1 |
| Protect | 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block<br>1 = Protective interlocking not activated | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| Prot_En | 1 = Protective interlock (protection, Protect parameter) is active | BOOL | 1 |
| PulseWidth* | Pulse width of control signal [s] | REAL | 3.0 |
| RapidStp* | Rapid stop for the motor:<br>0 = Motor On<br>1 = Motor Off | BOOL | 0 |
| RstLi* | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | 1 = Reset via operator | BOOL | 0 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SimAV* | Additional value used for SimOn = 1 | REAL | 0.0 |
| SimAV_Li | Additional analog value that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimLiOp | Activation/deactivation of the simulation<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | - |
| StartAut* | 1 = Starting the motor in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StartChnST | Signal status of output channel Start<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| StartForce | 1 = Force motor start | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StartLocal | 1 = Starting the motor in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StartMan* | 1 = Starting the motor in "manual mode" | BOOL | 0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| StopAut* | 1 = Stopping the motor in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopForce | 1 = Force motor stop | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopLocal | 1 = Stopping the motor in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopMan* | 1 = Stopping the motor in "manual mode" | BOOL | 0 |
| Trip | 1 = Motor is in "good" state | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| UserAna1 | Analog auxiliary value 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UA1unit | Unit of measure for analog auxiliary value 1 | INT | 0 |
| UserAna2 | Analog auxiliary value 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| UA2unit | Unit of measure for analog auxiliary value 2 | INT | 0 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |
| WarnTiAut | Prewarning of motor start in "automatic mode" in [s] | REAL | 0.0 |
| StateTrkOn | Initiate tracking of state | BOOL | 0 |
| WarnTiMan | Prewarning of motor start in "manual mode" in [s] | REAL | 0.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutAct | 1 = "Automatic mode" enabled<br>0 = "Manual mode" is enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutoST | 1 = Worst signal status of automatic commands | BYTE | 16#80 |
| AV_OpScale | Limit for scale in AV bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| AV_Out | Output additional analog value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| AV_Unit | Unit of measure for additional analog value | INT | 0 |
| CurrMon | Remaining monitoring time [s] | DINT | 0 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see MotL error handling (Page 1390) | INT | -1 |
| FaultExtOut | FaultExt as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| FbkRunOut | 1= Motor feedback running | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkStopOut | 1= Motor feedback stopped | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| GrpErr | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IntlockOut | Intlock as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalAct | 1 = "Local mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LockAct | 1 = Interlock (Intlock, Permit, Protect) or Trip is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManAct | 1 = "Manual mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| MonDynErr | 1 = Feedback error or feedback start error due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MonDynStopErr | 1 = Feedback stop error due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MonStaErr | 1 = Feedback error due to unexpected feedback change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message acknowledgement status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | Alarm error 1 (output ERROR of first ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1Perm | WORD | 16#FFFF |
| OS1PermLog | Display of OS1Perm with settings changed by the block algorithm | WORD | 16#FFFF |
| P_Rst | 1= Pulse output for reset<br>The parameter persists for one cycle after a reset. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Stop | 0 = Pulse signal for stopping the motor | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| P_Start | 1 = Pulse signal for starting the motor | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PermitOut | Permit as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PermitSafePosOut | PermitSafePos as output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| ProtectOut | Protect as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| R_StpAct | 1 = Rapid stop of the motor is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToReset | 1 = Ready for reset via RstLi input or commands in "local", "automatic", or "manual mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToStart | 1 = Active start readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Run | 1 = Motor is running | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Start | 1 = Control of motor: started | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Status1 | Status word 1 (Page 1376) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1376) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1376) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 1376) | DWORD | 16#00000000 |
| Stop | 1 = Motor stopped | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopCtrl | Control output to Stop the motor (go to Safe position) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| TripOut | Trip as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| UA1unitOut | UA1 as output | INT | 0 |
| UA2unitOut | UA2 as output | INT | 0 |
| UserAna1Out | UserAna1 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| UserAna2Out | UserAna2 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| WarnAct | 1 = Prewarning for motor start active (parameters WarnTiAut and WarnTiMan) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ResyncST | Worst Signal Status of inputs for resynchronization | BYTE | 16#80 |

## See also

MotL modes (Page 1380)

MotL block diagram (Page 1404)

MotL messaging (Page 1392)

## 8.2.7 MotL block diagram

### MotL block diagram

A block diagram is not provided for this block.

### See also

MotL I/Os (Page 1394)

MotL messaging (Page 1392)

MotL error handling (Page 1390)

MotL functions (Page 1382)

MotL modes (Page 1380)

Description of MotL (Page 1376)

## 8.2.8 Operator control and monitoring

### 8.2.8.1 MotL views

### Views of the MotL block

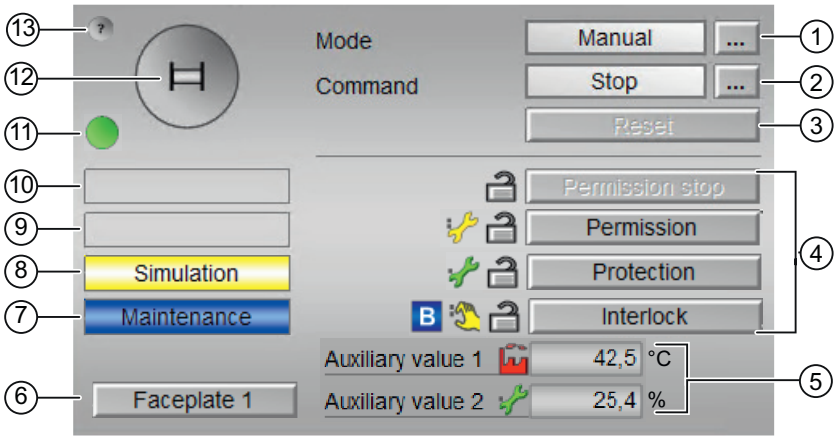
The MotL block provides the following views:

- MotL standard view (Page 1405)
- Alarm view (Page 336)
- Limit view of motors (Page 328)
- Trend view (Page 341)
- Parameter view for motors and valves (Page 320)
- MotL preview (Page 1408)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for MotL (Page 1412)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 8.2.8.2 MotL standard view

#### MotLstandard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual and automatic mode for motors, valves and dosers (Page 82)
- Manual and automatic mode for motors, valves and dosers (Page 82)
- Local mode (Page 86)
- Out of service (Page 77)

Refer to section Switching operating states and operating modes (Page 289) for information on switching the operating mode.

#### (2) Starting and stopping the motor

This area shows you the default operating state for the motor. The following states can be shown and executed here:

- "Start"
- "Stop"
- "Rapid stop"

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

If the text is configured for these commands, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (3) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in the section Resetting the block in case of interlocks or errors (Page 47).

### (4) Operator control and display area for interlock functions of the block

This display is only visible when the corresponding block input is connected.

You can use this button to control the interlock functions of the block. You can find additional information on this in the section Interlocking functions (Page 107).

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 113)), for example:

- Signal status (see Forming and outputting the signal status for technologic blocks (Page 117)), for example:

- Bypass information (see Forming the group status for interlock information (Page 113)):

If the text is configured for the interlocks, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (5) Display of auxiliary values

This display is only visible when the corresponding block input is connected.

You can use this area to display two auxiliary values that have been configured in the engineering system (ES). You can find additional information on this in the section Displaying auxiliary values (Page 239).

### (6) Navigation button for switching to the standard view of any faceplate

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234).

### (7) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this is available in section Release for maintenance (Page 70) Display area for block states.

### (8) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"
- "Delay"

You will find more detailed information on this in the sections Simulating signals (Page 64) and Display of delay times (Page 288).

### (9) Display area for block states

This area provides additional information on the operating state of the block (from high to low according to priority):

- "Motor protection"
- "External error"
- "Status error"
- "Control error"
- "Internal error"
- "Changeover error"

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 104) , Error handling (Page 128) (section "Invalid input signals" and "Mode switchover error") and Motor protection function (Page 106).

### (10) Display area for block states

This area provides additional information on the operating state of the block:

- "Forced start"
- "Forced stop"
- "Request 0/1": Reset is expected but reset is not possible from faceplate. For more details, refer:Reset of errors during Request 0/1 (Page 382)

You can find additional information on this in the section Forcing operating modes (Page 45).

### (11) Automatic preview

This display is only visible in "manual mode", in "local mode", or with a reset request in "automatic mode", when the current output signals are not identical to the control in "automatic mode".

The display shows what state the motor would assume if you switched from "manual" or "local" mode to "automatic mode", or performed a reset to "automatic mode".

### (12) Status display of the motor

The current status of the motor is graphically displayed here.

You can find more information about this in section Block icon for MotL (Page 1412).

### (13) Mini preview for Resynchronization

Displays the mini preview for Resynchronizing the signal state.

The mini preview displays the following states:

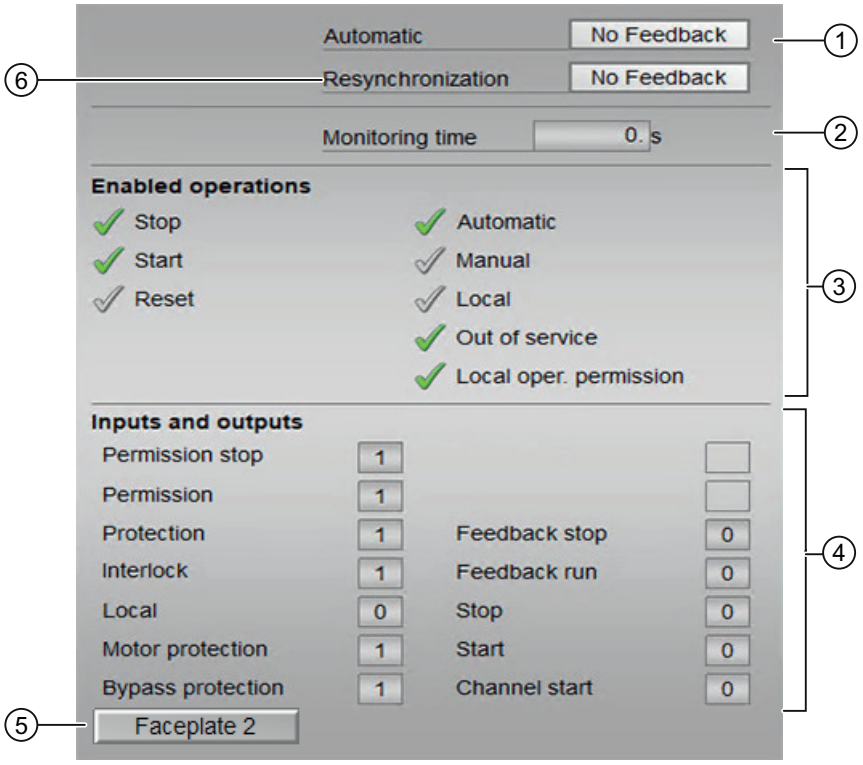
- Start
- Stop
- No Cmd

**Note**

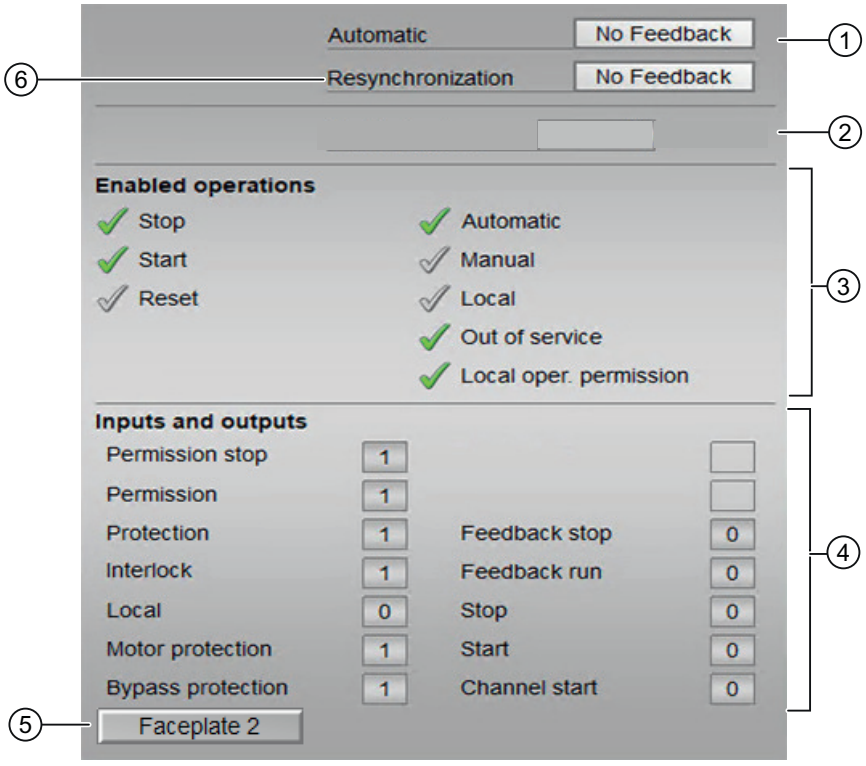
Mini preview for resynch is visible only if resynchronization command is different to control output value.

## 8.2.8.3 MotL preview

### Preview of MotL



Display of the current monitoring time is visible.




Display of the remaining monitoring time is not visible.

## (1) Automatic preview

This area shows you the block status after it has switched from the "Manual" mode or "Local" mode to the "Automatic" mode.

If the block is in "Automatic" mode, the current block state is displayed.

**Note**

- If the valve is closed and interlock locking is active (`SafePos`=0 Close), the automatic preview does not show the current status "Closed", instead it shows the active automatic command "Open".
- If the valve is in the end position and the automatic stop command is active, the automatic preview does not show the current status "Closed", but the active automatic command "Stop".

This area also displays the worst signal status of the following automatic commands:

- `StartAut`
- `StopAut`

### (2) Monitoring time

The remaining "Monitoring time" is displayed in this area.

### (3) Enable operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "Stop": You can stop the motor.
  If the text is configured for this command, it is also displayed in brackets. For more informtaion, see section Labeling of buttons and text (Page 235).
- "Start": You can start the motor.
  If the text is configured for this command, it is also displayed in brackets. For more informtaion, see section Labeling of buttons and text (Page 235).
- "Reset": You can reset the motor after interlocks or errors.
- "Automatic": You can switch to "automatic mode".
- "Manual": You can switch to "manual mode".
- "Local": You can switch to "local mode".
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operator permission": Use the ← button to switch to the standard view of the OpStations block. You can find additional information on this in the section Operator control permissions (Page 285).
- "Monitoring time": Display of the remaining monitoring time.

### (4) Displaying current control signals

This area shows the most important parameters for this block with the current selection:

- "Permission":
  This display is only visible when the corresponding block input is connected.
  - 0 = No OS release for energizing motor
  - 1 = Enable for "starting"/"stopping" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection":
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock":
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Local": 1 = Control signal for "Local mode" (LocalLi) is active
- "Motor protection": 1 = Motor is in "good" state
- "Interlock deact.":
  - 0 = Bypass disabled
  - 1 = Bypassing interlock in "local mode" and in "simulation"
- "Local stop": 1 = Stopping the motor in "local mode"
- "Local start": 1 = Starting the motor in "local mode"
- "Feedback run →": 1 = Motor has started and is running
- "Feedback stop" : 1 = Motor has stopped
- "Stop": 1 = Stop motor
- "Start": 1 = Start motor
- "Channel Start": Signal from the output channel block "Start"
- "Permission stop": 1 = Motor can not be stopped in manual (by operator) or automatic (by logic) mode

### (5) Navigation button for switching to the standard view of any faceplate

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

For more informtaion, see section Opening additional faceplates (Page 234).

### (6) Resynchronization

Displays the Resynchronization state and Resynchronization Signal status based on feedback value.

## 8.2.8.4 Block icon for MotL

### Block symbols for MotL

A variety of block symbols are available with the following functions:

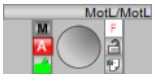
- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as control system faults
- Operating modes
- Signal status, release for maintenance
- Displays for bypassing interlocks
- Interlocks
- Memo display
- Motor state display

The block symbols from template @TemplateAPLV9.PDL:

| Symbols | Selection of the block symbol in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | |

| Symbols | Selection of the block symbol in CFC | Special features |
|---|---|---|
|  | 4 | |
|  | 5 | |
|  | 6 | |

The block symbols from template @TemplateAPLV7.PDL:

| Symbols | Selection of the block symbol in CFC | Special features |
|---|---|---|
|  | 1 | Block symbol in the full display |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | |
|  | 6 | |
|  | 7 | |
|  | 8 | |
|  | 9 | |
|  | 10 | |
|  | - | Block symbol in "Out of service" mode (example with type 1 block symbol) |

Additional information on the block symbol and the control options in the block symbol is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

## Motor state display

The following motor states are shown here:

| Symbol | Meaning |
|---|---|
|  | Motor started (motor symbol changes) |
|  | The motor is running |
|  | Motor stopped (motor symbol changes) |
|  | Motor idle |
|  | Error at motor (monitoring error, motor protection) |
|  | Motor out of service |
|  | “No Cmd” State |

# 8.3 MotS - Motor (Small)

## 8.3.1 Description of MotS

### Object name (type + number)

Type + number: FB 1910

Family: Drives

### Area of application for MotS

The block is used for the following applications:

- Controlling motors

**Note**

This block is also available as a large block. A comparison of the MotL and MotS blocks is available in the section: MotL compared to MotS (Page 1369)

### How it works

The block is used to control motors. Various inputs are available for controlling the motor.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

### Status word allocation

For a description of the individual parameters, see the section MotS I/Os (Page 1426).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |

| Status bit | Parameter |
|---|---|
| 4 | `OosLi.Value` |
| 5 | 0 = `ManAct.Value`<br>1 = `AutoAct.Value` |
| 6 | `LocalAct.Value` |
| 7 | `LockAct.Value` |
| 8 | `Start.Value` |
| 9 | Motor is stopped |
| 10 | Not used |
| 11 | `MonStaErr.Value` |
| 12 | `MonDynErr.Value` |
| 13 | `BypProt` |
| 14 | Invalid signal status |
| 15 | Not used |
| 16 | 1 = `Intlock` is active |
| 17 - 18 | Not used |
| 19 | `Trip` |
| 20 - 22 | Not used |
| 23 | "Interlock" button is enabled |
| 24 | Reset request in automatic preview |
| 25 | Not used |
| 26 | Bypass information for display in block icon |
| 27 | Automatic preview for "starting" |
| 28 | Automatic preview for "stopping" |
| 29 | External error generated by FaultExt or external control system fault CSF with set `Feature.Bit18` Activating error state for external process control error CSF (Page 162) |
| 30 - 31 | Not used |

## Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `MsgLock` |
| 1 - 18 | Not used |
| 19 | 1 = No impact of input signals on "local mode" when `LocalSetting` = 0, `LocalSetting` = 2 and OOS mode. |
| 20 | Motor is stopped |
| 21 | Motor is stopping |
| 22 | Motor is starting |
| 23 | Motor is running |
| 24 | Error in motor |
| 25 | 1 = Input parameter `StartChnST` is interconnected |
| 26 - 29 | Not used |

| Status bit | Parameter |
|---|---|
| 30 | Display for interlocks in block icon |
| 31 | `MS_RelOp` |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 - 11 | Not used |
| 12 | Hidden bypass signal in interlock |
| 13 | Not used |
| 14 | `Feature2.Bit2`: Separate bypass signal |
| 15 - 23 | Not used |
| 24 | Output command for "starting" the motor |
| 25 | Output command for "stopping" the motor |
| 26 | Show automatic preview in the standard view |
| 27 | Internal error |
| 28 | `GrpErr.Value` |
| 29 | Enable for "starting" the motor |
| 30 | Not used |
| 31 | Not used |

### Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 - 8 | Not used |
| 9 | `Trip` not connected |
| 10 | Do not connect `FbkOutRun` |
| 11 - 31 | Not used |

### See also

MotS messaging (Page 1425)
MotS block diagram (Page 1432)
MotS functions (Page 1419)
MotS error handling (Page 1424)
MotS modes (Page 1418)

## 8.3.2 MotS modes

### MotS operating modes

The block can be operated using the following modes:

- Local mode (Page 86)
- Automatic mode (Page 82)
- Manual mode (Page 82)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

### "Local mode"

You can find general information on "Local mode", switching modes and Bumpless switchover in the Local mode (Page 86) section.

**Note**

**"Local mode" for the block MotS**

In contrast to the "Large" blocks, it is only possible to perform settings in this block `LocalSetting` with 0 or 2. A "local operation" is accordingly possible only via the internal tracking of the feedback value.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and Bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 82) section.

Motor actions you can control in auto mode:

- "Start" (`StartAut` = 1)
- "Stop" (`StopAut` = 1)

**Note**

**Information about the "Small" block**

This "Small" block works with pushbutton operation. The automatic commands are therefore latching, in other words, `OpenAut`, `CloseAut` can be reset to 0 after the control is changed. In "manual" and "local" modes, however, the automatic commands are not saved and in the absence of automatic commands the automatic control is tracked.

### "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the section Manual and automatic mode for motors, valves and dosers (Page 82).

Motor actions you can control in "manual mode":

- "Start" (`StartMan` = 1)
- "Stop" (`StopMan` = 1)

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature2.Bit1` =0 and `Feature2.Bit1` =1 is as shown below:

| Output Parameters | **`Feature2.Bit1` =0** | **`Feature2.Bit1` =1** |
|---|---|---|
| FbkRunOut.ST | 16#80 | 16#60 |
| Run.ST | 16#80 | 16#60 |
| Stop.ST | 16#80 | 16#60 |

### See also

Description of MotS (Page 1415)

MotS functions (Page 1419)

MotS error handling (Page 1424)

MotS messaging (Page 1425)

MotS I/Os (Page 1426)

MotS block diagram (Page 1432)

## 8.3.3 MotS functions

### Functions of MotS

The functions for this block are listed below.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" |
| 1 | 1 = Operator can switch to "manual mode" |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can stop the motor |
| 5 | 1 = Operator can start the motor |
| 6 | Not used |
| 7 | 1 = Operator can reset the motor |
| 8 | 1 = Operator can define the monitoring time for startup |
| 9 | Not used |
| 10 | 1 = Operator can activate the monitoring time function (Bit 8) |
| 11 | 1 = Operator can enable functional simulation |
| 12 | 1 = Operator can activate the Release for maintenance function |
| 13 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

## Button labels

This block provides the standard function Labeling of buttons and text (Page 235).

Instance-specific text can be configured for the following parameters:

- `StartMan`
- `StopMan`

## Interlocks

This block provides the following interlocks:

- Interlock without reset (interlock)

Refer to the section Interlocks (Page 107) as well as Influence of the signal status on the interlock (Page 112).

## Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Motor protection function

This block provides the standard function Motor protection function (Page 106).

### Disabling interlocks

This block provides the standard function Disabling interlocks (Page 111).

### Resetting the block in case of interlocks or errors

This block provides the standard function Resetting the block in case of interlocks or errors (Page 47).

### External error (`FaultExt`), external control system fault (`CSF`)

This block provides the possibility to pass an external error via the `FaultExt` parameter or an external control system fault via the CSF parameter. See MotS error handling (Page 1424).

### Group error

This block provides the standard function Outputting group errors (Page 130).

The following parameters are taken into consideration when forming the group error:

- `CSF`
- `Trip`
- `MonDynErr`
- `MonStaErr`
- `FaultExt`

### Forming the group status for interlocks

This block provides the standard function Forming the group status for interlock information (Page 113).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `LocalLi.ST`
- `Trip.ST`
- `FbkRunOut.ST`
- `StartChn.ST`

- `StartAut.ST` (only if `Feature2.Bit10` = 1)
- `StopAut.ST` (only if `Feature2.Bit10` = 1)

### Considering bad quality of automatic commands or external values

If the `Feature2` bit Considering bad quality of automatic commands or external values (Page 200) is set to 1 and one of the following parameters has bad signal status (16#00 or 16#28), the block goes to the neutral position (motor stop) in the "Automatic" mode:

- `StartAut.ST`
- `StopAut.ST`

### Outputting a signal for start readiness

This block provides the standard function Outputting a signal for start readiness (Page 59).

### Feedback monitoring

This block provides the standard function Monitoring the feedbacks (Page 104).

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

### Neutral position

This block provides the standard function Neutral position for motors, valves and controllers (Page 54).

### Output signal as a static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 57).

### Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 230).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the section Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |

| Bit | Function |
|---|---|
| 2 | Resetting the commands for changing the mode (Page 172) |
| 3 | Enabling resetting of commands for the control settings (Page 172) |
| 9 | Resetting via input signals in the event of interlocking (Protection) or errors (Page 174) |
| 10 | Exiting local mode (Page 189) |
| 11 | Activating the run time of feedback signals (Page 164) |
| 18 | Activating error state for external process control error CSF (Page 162) |
| 19 | Reset even with locked state (Page 176) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 27 | Interlock display with LocalSetting 2 or 4 (Page 189) |
| 30 | Set reset depending on the operating mode or the LiOp parameter (Page 174) |
| 31 | Activating reset of protection / error in manual mode (Page 175) |

### Configurable reactions using the `Feature2` parameter

| Bit | Function |
|---|---|
| 1 | Changes signal status of outputs in OOS (Page 208) |
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 3 | Control priority in the event of invalid or not plausible input commands (Page 186) |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) |
| 10 | Considering bad quality of automatic commands or external values (Page 200) |

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### See also

Description of MotS (Page 1415)

MotS messaging (Page 1425)

MotS I/Os (Page 1426)

MotS block diagram (Page 1432)

MotS modes (Page 1418)

Selecting a unit of measure (Page 240)

## 8.3.4 MotS error handling

### Error handling of MotS

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Mode switchover error
- Control system fault (CSF)
- Invalid input signals

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers.

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed. |
| 0 | There is no error. |
| 41 | The value for the `LocalSetting` I/O is not 0 or 2 |
| 42 | `LocalSetting` = 0 and `LocalLi` = 1 |
| 51 | `StartAut` = 1 and `StopAut` = 1, `AutModLi` = 1 and `ManModLi` = 1 |
| 52 | `LocalAct` = 1 and `LocalSetting` = 2 and `SimOn` = 1 |

### Mode switchover error

This error can be output by the block, see the section Error handling (Page 128).

### Invalid input signals

This error can be output by the block, see the section Error handling (Page 128).

For the following invalid input signals, the control output can be kept or switched to the neutral position. This depends on the function Control priority in the event of invalid or not plausible input commands (Page 186).

| Operating mode | Invalid input signals | Control reaction with `Feature2.Bit3` =1 |
|---|---|---|
| "Automatic mode" | `StartAut` = 1 and `StopAut` = 1 | Motor is stopping |
| "Manual mode" | `StopMan` = 1 and `StartMan` = 1 | |

### Control system fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 128) section for more on this.

### See also

MotS functions (Page 1419)

MotS messaging (Page 1425)

MotS I/Os (Page 1426)

Description of MotS (Page 1415)

MotS modes (Page 1418)

MotS block diagram (Page 1432)

## 8.3.5 MotS messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Instance-specific messages

### Process control fault

The following control system error messages can be output:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 1 | AS process control message - fault | $$BlockComment$$<br>Motor feedback error |
| | SIG 2 | AS process control message - fault | $$BlockComment$$<br>Motor protection triggered |
| | SIG 3 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If it changes to CSF = 1, a process control fault is triggered (MsgEvId1, SIG 3).

### Instance-specific messages

You have the option to use two instance-specific messages for this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External message 1 |
| | SIG 5 | AS process control message - fault | $$BlockComment$$<br>External message 2 |

Explanation:

$$BlockComment$$: Content of the instance-specific comment.

### Associated values for message instance **MsgEvId1**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |
| 4 | ExtVa104 |
| 5 | ExtVa105 |
| 6 - 10 | Reserved |

The associated values 4 ... 5 are allocated to the parameters ExtVa104 ... ExtVa105 and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of MotS (Page 1415)

MotS functions (Page 1419)

MotS error handling (Page 1424)

MotS block diagram (Page 1432)

MotS I/Os (Page 1426)

MotS modes (Page 1418)

## 8.3.6 MotS I/Os

I/Os of MotS

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

## Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutModLi* | 1 = "Automatic mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutModOp* | 1 = "Automatic mode" via operator (controlled by ModLiOp = 0) | BOOL | 0 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BypProt | 1 = Bypassing interlock in "local mode" and in "simulation" | BOOL | 0 |
| CSF | 1 = External error (control system fault) Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa104 | Associated value 4 for messages (MsgEvID1) | ANY | |
| ExtVa105 | Associated value 5 for messages (MsgEvID1) | ANY | |
| FaultExt | 1 = External error<br>Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkRun | Feedback for starting is present:<br>1 = Start<br>0 = Stop | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Feature | I/O for additional functions (Page 1419) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| Feature2 | I/O for additional functions (Page 1419) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| Intlock | 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared<br>1 = Interlock not activated | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| Intl_En | 1 = Interlock without reset (interlock, Intlock parameter) is active | BOOL | 1 |
| LocalLi | 1 = Activate "local mode" via plant signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalSetting | Properties for the Local mode (Page 86) | INT | 0 |
| ManModLi* | 1 = "Manual mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManModOp* | 1 = "Manual mode" via OS operator (controlled by ModLiOp = 0) | BOOL | 1 |
| ModLiOp | Switchover of operating mode between:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Monitor | 1 = Feedback monitoring | BOOL | 1 |
| MonTiDynamic* | Monitoring time for feedback errors after operation in [s] | REAL | 3.0 |
| MS_RelOp* | 1= Release for maintenance via OS operator | BOOL | 0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = In use by a batch | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 1419) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| RstLi* | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | 1 = Reset via operator | BOOL | 0 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SimOn | 1 = Simulation on | BOOL | 0 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| StartAut* | 1 = Starting the motor in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StartChnST | Signal status of output channel Start<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| StartMan* | 1 = Starting the motor in "manual mode" | BOOL | 0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| StopAut* | 1 = Stopping the motor in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopMan* | 1 = Stopping the motor in "manual mode" | BOOL | 0 |
| Trip | 1 = Motor is in "good" state | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutAct | 1 = "Automatic mode" enabled<br>0 = "Manual mode" is enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutoST | 1 = Worst signal status of automatic commands | BYTE | 16#80 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see MotS error handling (Page 1424) | INT | -1 |
| FaultExtOut | FaultExt as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkRunOut | Feedback for starting is present:<br>1 = Start<br>0 = Stop | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| GrpErr | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IntlockOut | Intlock as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalAct | 1 = "Local mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LockAct | 1 = Interlock or Trip is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManAct | 1 = "Manual mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| MonDynErr | 1 = Feedback error due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MonStaErr | 1 = Feedback error due to unexpected feedback change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Rst | 1= Pulse output for reset<br>The parameter persists for one cycle after a reset. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| RdyToReset | 1 = Ready for reset via RstLi input or commands in "local", "automatic", or "manual mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToStart | 1 = Active start readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Run | 1 = Motor is running | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Start | 1 = Control of motor: started | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Status1 | Status word 1 (Page 1415) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1415) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1415) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 1415) | DWORD | 16#00000000 |
| Stop | 1 = Motor stopped | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| TripOut | Trip as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

**See also**

MotS messaging (Page 1425)
MotS block diagram (Page 1432)
MotS modes (Page 1418)
Error handling (Page 128)

## 8.3.7 MotS block diagram

**MotS block diagram**

A block diagram is not provided for this block.

**See also**

Description of MotS (Page 1415)
MotS functions (Page 1419)
MotS messaging (Page 1425)
MotS I/Os (Page 1426)
MotS modes (Page 1418)
MotS error handling (Page 1424)

## 8.3.8 Operator control and monitoring

### 8.3.8.1 MotS views

**Views of the MotS block**

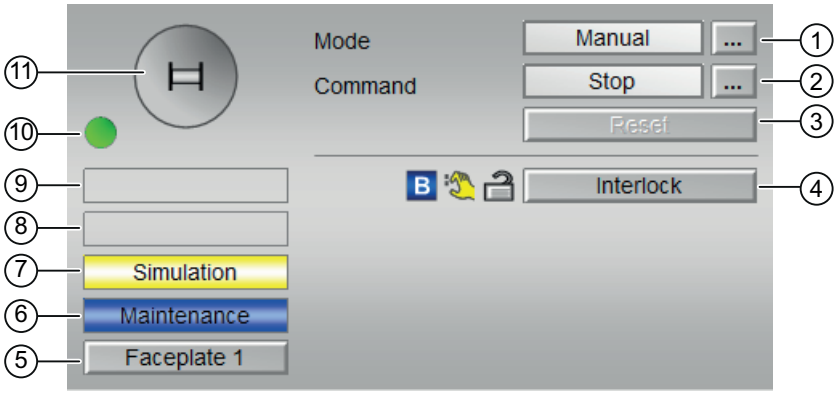
The block MotS provides the following views:

- MotS standard view (Page 1433)
- Alarm view (Page 336)
- Trend view (Page 341)
- Parameter view for motors and valves (Page 320)
- MotS preview (Page 1436)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for MotS (Page 1438)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 8.3.8.2 MotS standard view

#### MotSstandard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual and automatic mode for motors, valves and dosers (Page 82)
- Manual and automatic mode for motors, valves and dosers (Page 82)
- Local mode (Page 86)
- Out of service (Page 77)

Refer to section Switching operating states and operating modes (Page 289) for information on switching the operating mode.

#### (2) Starting and stopping the motor

This area shows you the default operating state for the motor. The following states can be shown and executed here:

- "Start"
- "Stop"

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

If the text is configured for these commands, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (3) Resetting the block

Click "Reset" for errors. You can find additional information on this in the section Resetting the block in case of interlocks or errors (Page 47).

### (4) Operator control and display area for the interlock function of the block

This display is only visible when the corresponding block input is connected.

You can use this button to control the interlock function of the block. You can find additional information on this in the section Interlocks (Page 107).

The following is displayed in addition to the button:

- Interlock status (see Forming the group status for interlock information (Page 113)), for example:

- Signal status (see Forming and outputting the signal status for technologic blocks (Page 117)), for example:

- Bypass information (see Forming the group status for interlock information (Page 113)):

If the text is configured for the interlocks, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (5) Navigation button for switching to the standard view of any faceplate

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234).

### (6) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this is available in section Release for maintenance (Page 70) Display area for block states.

### (7) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"

You can find additional information on this in the section Simulating signals (Page 64).

### (8) Display area for block states

This area provides additional information on the operating state of the block (from high to low according to priority):

- "Motor protection"
- "External error"
- "Status error"
- "Control error"
- "Internal error"
- "Invalid Signal"

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 104) , Error handling (Page 128) (subsection "Invalid input signals") and Motor protection function (Page 106).

### (9) Display area for block states

This area provides additional information on the operating state of the block:

- "Request 0/1": Reset is expected but reset is not possible from faceplate. For more details, refer: Reset of errors during Request 0/1 (Page 382)

### (10) Automatic preview

This display is only visible in "manual mode", in "local mode", or with a reset request in "automatic mode", when the current output signals are not identical to the control in "automatic mode".

The display shows what state the motor would assume if you switched from "manual" or "local" mode to "automatic mode", or performed a reset to "automatic mode".

### (11) Status display of the motor

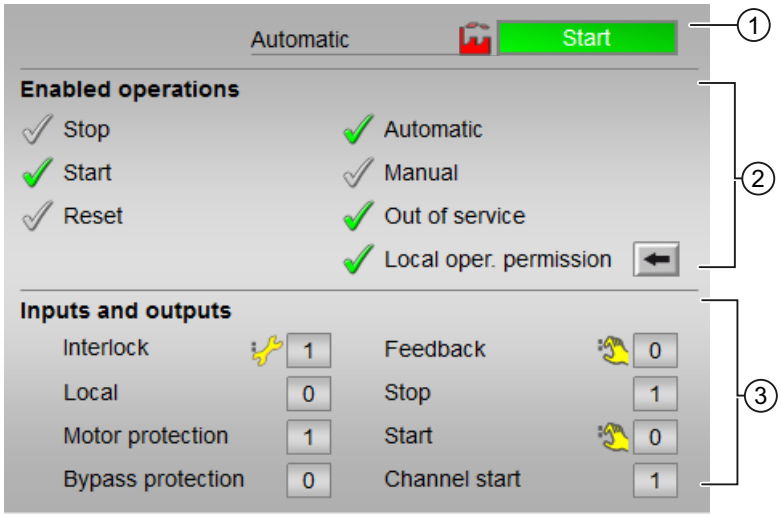
The current status of the motor is graphically displayed here:

- Green: Motor is running
- Gray: Motor idle
- Red: A fault has occurred

You can find additional information on this in the section Block icon for MotS (Page 1438).

### 8.3.8.3 MotS preview

#### Preview of MotS




#### (1) Automatic preview

This area shows you the block status after it has switched from the "Manual" mode or "Local" mode to the "Automatic" mode.

If the block is in "Automatic" mode, the current block state is displayed.

**Note**

- If the valve is closed and interlock locking is active (`SafePos`=0 Close), the automatic preview does not show the current status "Closed", instead it shows the active automatic command "Open".
- If the valve is in the end position and the automatic stop command is active, the automatic preview does not show the current status "Closed", but the active automatic command "Stop".

This area also displays the worst signal status of the following automatic commands:

- `StartAut`
- `StopAut`

#### (2) Enable operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "Stop": You can stop the motor.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Start": You can start the motor.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Reset": You can reset the motor after interlocks or errors.
- "Automatic": You can switch to "automatic mode".
- "Manual": You can switch to "manual mode".
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operator permission": Use the ← button to switch to the standard view of the OpStations block. You can find additional information on this in the section sedtion Operator control permissions (Page 285).

### (3) Displaying current control signals

This area shows the most important parameters for this block with the current selection:

- "Interlock":
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Local": 1 = Control signal for "Local mode" (LocalLi) is active
- "Motor protection": 1 = Motor is in "good" state
- "Interlock deact.":
  - 0 = Bypass disabled
  - 1 = Bypassing interlock in "local mode" and in "simulation"
- "Feedback →": 1 = Motor has started and is running
- "Stop": 1 = Stop motor
- "Start": 1 = Start motor
- "Channel Start": Signal from the output channel block for "Start"

### 8.3.8.4 Block icon for MotS

#### Block symbols for MotS

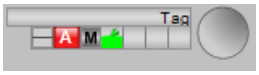
A variety of block symbols are available with the following functions:

- Process tag type
- Violation of alarm, warning, and tolerance limits as well as control system faults
- Operating modes
- Signal status, release for maintenance
- Displays for bypassing interlocks
- Interlocks
- Memo display
- Motor state display

The block symbols from template @TemplateAPLV9.PDL:

| Symbols | Selection of the block symbol in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | |
|  | 6 | |

The block symbols from template @TemplateAPLV7.PDL:

| Symbols | Selection of the block symbol in CFC | Special features |
|---|---|---|
|  | 1 | Block symbol in the full display |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | |
|  | 6 | |
|  | - | Block symbol in "Out of service" mode (example with type 1 block symbol) |

Additional information on the block symbol and the control options in the block symbol is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

## Motor state display

The following motor states are shown here:

| Symbol | Meaning |
|---|---|
|  | Motor started (motor symbol changes) |
| | The motor is running |
| | Motor stopped (motor symbol changes) |
| | Motor idle |
|  | Error at motor (monitoring error, motor protection) |

# 8.4 MotRevL - Reversible motor

## 8.4.1 Description of MotRevL

### Object name (type + number) and family

Type + number: FB 1851

Family: Drives

### Area of application for MotRevL

The block is used for the following applications:

- Control of reversible motors

### How it works

The block is used to control reversible motors. Various inputs are available for controlling the motor.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

For the MotRevL block, the Advanced Process Library contains a template for process tag types as an example with an application scenario for this block.

Example of process tag types:

- Reversing motor (MotorReversible) (Page 2861)

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section MotRevL I/Os (Page 1463).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |

| Status bit | Parameter |
|---|---|
| 4 | OosLi.Value |
| 5 | 0 = ManAct.Value<br>1 = AutoAct.Value |
| 6 | LocalAct.Value |
| 7 | 0: Open padlock in the block icon<br>1: Closed padlock in the block icon |
| 8 | Fwd.Value |
| 9 | Motor is stopped |
| 10 | Rev.Value |
| 11 | MonStaErr.Value |
| 12 | MonDynErr.Value |
| 13 | BypProt |
| 14 | Invalid signal status |
| 15 | Mode switchover error |
| 16 | 1 = Intlock is active |
| 17 | 1 = Permit is active |
| 18 | 1 = Protect is active |
| 19 | Trip.Value |
| 20 | Display „Forced forward“ |
| 21 | Display „Forced stop“ |
| 22 | Display „Forced reverse“ |
| 23 | "Interlock" button is enabled |
| 24 | Reset request in automatic preview |
| 25 | WarnAct.Value or<br>IdleTime active |
| 26 | Bypass information for display in block icon |
| 27 | Automatic preview for forward mode |
| 28 | Automatic preview for "stopping" |
| 29 | Automatic preview for reverse mode |
| 30 | "Permission" button is enabled |
| 31 | "Protection" button is enabled |

### Status word allocation for Status2 parameter

| Status bit | Parameter |
|---|---|
| 0 | MsgLock |
| 1 | AV_AH_Act.Value |
| 2 | AV_WH_Act.Value |
| 3 | AV_TH_Act.Value |
| 4 | AV_TL_Act.Value |
| 5 | AV_WL_Act.Value |
| 6 | AV_AL_Act.Value |

| Status bit | Parameter |
|---|---|
| 7 | AV_AH_En OR Feature.Bit23 |
| 8 | AV_WH_En OR Feature.Bit23 |
| 9 | AV_TH_En OR Feature.Bit23 |
| 10 | AV_TL_En OR Feature.Bit23 |
| 11 | AV_WL_En OR Feature.Bit23 |
| 12 | AV_AL_En OR Feature.Bit23 |
| 13 | AV_AH_MsgEn |
| 14 | AV_WH_MsgEn |
| 15 | AV_TH_MsgEn |
| 16 | AV_TL_MsgEn |
| 17 | AV_WL_MsgEn |
| 18 | AV_AL_MsgEn |
| 19 | 1 = No impact of input signals on "local mode" with LocalSetting = 0, LocalSetting = 2, LocalSetting = 4 and OOS mode. |
| 20 | Motor is stopped |
| 21 | Motor stops in forward mode |
| 22 | Motor stops in reverse mode |
| 23 | Motor starts in forward mode |
| 24 | Motor runs in forward mode |
| 25 | Motor starts in reverse mode |
| 26 | Motor runs in reverse mode |
| 27 | Error when stopping motor |
| 28 | Error in forward mode of motor |
| 29 | Error in reverse mode of motor |
| 30 | Display for interlocks in block icon |
| 31 | MS_RelOp |

### Status word allocation for Status3 parameter

| Status bit | Parameter |
|---|---|
| 0 | Delay of the AV_AH_Lim message |
| 1 | Delay of the AV_WH_Lim message |
| 2 | Delay of the AV_TH_Lim message |
| 3 | Delay of the AV_TL_Lim message |
| 4 | Delay of the AV_WL_Lim message |
| 5 | Delay of the AV_AL_Lim message |
| 6 | Collection of message delays |
| 7 | Resynchronization preview for Forward |
| 8 | Resynchronization preview for Reverse |
| 9 | Resynchronization preview for Stop |
| 10 | Resynchronization: Display preview |
| 11 | Resynchronization: Display small preview symbol in standard view |

| Status bit | Parameter |
|---|---|
| 12 | 1 = "Permission stop" button is enabled |
| 13 | 1 = Permission stop is active |
| 14 | Resynchronization with state “No Cmd” |
| 15 | Current monitoring time is visible |
| 16 | `MonDynStopErr.Value` |
| 17 | Not used |
| 18 | `SimLiOp.Value` |
| 19 | 1 = Enable for rapid stop (`Feature` Bit Enabling rapid stop via faceplate (Page 179)) |
| 20 | 1 = Input parameter `FwdChnST` is interconnected |
| 21 | 1 = Input parameter `RevChnST` is interconnected |
| 22 | Not used |
| 23 | Command for rapid stop |
| 24 | Command for starting → the motor |
| 25 | Command for starting ← the motor |
| 26 | Show automatic preview in the standard view |
| 27 | Internal error |
| 28 | `GrpErr.Value` |
| 29 | `RdyToStart.Value` |
| 30 | Auxiliary value 1 visible |
| 31 | Auxiliary value 2 visible |

### Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 | Effective signal 1 of the message block connected via `EventTsIn` |
| 1 | Effective signal 2 of the message block connected via `EventTsIn` |
| 2 | Effective signal `3` of the message block connected via `EventTsIn` |
| 3 | Effective signal 4 of the message block connected via `EventTsIn` |
| 4 | Effective signal 5 of the message block connected via `EventTsIn` |
| 5 | Effective signal 6 of the message block connected via `EventTsIn` |
| 6 | Effective signal 7 of the message block connected via `EventTsIn` |
| 7 | Effective signal 8 of the message block connected via `EventTsIn` |
| 8 | `AV` not connected |
| 9 | Motor protection display (`Trip.Status` ≠ 16#FF) |
| 10 | 1 = Input parameter `FbkFwd` is connected |
| 11 | 1 = Input parameter `FbkRev` is connected |
| 12 - 19 | Effective signal 1...8 of the message block connected via `EventTs2In` |
| 20 | 1 = Input parameter `FbkStop` is connected |
| 21 | Not used |
| 22 | External error generated by FaultExt or external control system fault from CSF with set `Feature.Bit18` Activating error state for external process control error CSF (Page 162) |
| 23 | Hidden bypass signal in Permit |

| Status bit | Parameter |
|---|---|
| 24 | Hidden bypass signal in interlock |
| 25 | Hidden bypass signal in Protect |
| 26 | `Feature2.Bit2`: Separate bypass signal |
| 27 | `Feature2.Bit16`: Separate interlock for each direction or position |
| 28 | Hidden bypass signal in Permit Reverse |
| 29 | Hidden bypass signal in Interlock Reverse |
| 30 | Hidden bypass signal in Protect Reverse |
| 31 | Separate monitoring of shutdown of the motor (`Feature.Bit13`) |

### Status word allocation for `Status5` parameter

| Status bit | Parameter |
|---|---|
| 0 - 7 | Effective signal 9...16 of the message block connected via `EventTsIn` |
| 8 - 15 | Effective signal 9...16 of the message block connected via `EventTs2In` |
| 16 | The permission button for "Forward" is enabled |
| 17 | The permission button for "Reverse" is enabled |
| 18 | The interlock button for "Forward" is enabled |
| 19 | The Interlock button for "Reverse" is enabled |
| 20 | The protection button "Forward" is enabled |
| 21 | The protection button "Reverse" is enabled |
| 22 | 1 = Permission button "Reverse" is active |
| 23 | 1 = Interlock button "Reverse" is active |
| 24 | 1 = Protection button "Reverse" is active |
| 25 - 31 | Not used |
| 29 | 1= Suppressed messages |
| 30 - 31 | Not used |

### See also

MotRevL functions (Page 1447)
MotRevL messaging (Page 1461)
MotRevL block diagram (Page 1474)
MotRevL error handling (Page 1457)
MotRevL modes (Page 1445)

## 8.4.2 MotRevL modes

### MotRevL operating modes

The block can be operated using the following modes:

- Local mode (Page 86)
- Automatic mode (Page 82)
- Manual mode (Page 82)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

### "Local mode"

You can find general information on "Local mode", switching modes and Bumpless switchover in the Local mode (Page 86) section.

Motor actions you can control in local mode:

- "Starting in forward" (`FwdLocal` = 1)
- "Starting in reverse" (`RevLocal` = 1)
- "Stopping" (`StopLocal` = 1)

A motor operated in "local mode" is controlled either by "local" signals or by the feedback signals (input parameters `FbkFwd` = 1, `FbkRev` = 1 or `FbkStop` =1) . Configuration takes place via the input parameter `LocalSetting`.

`LocalSetting` =2,4: The control outputs are tracked to the feedbacks.

All three feedbacks are available and plausible means that always only one feedback is active. The corresponding control output is tracked:

```
Fwd = FbkFwd,
Rev = FbkRev,
StopCtrl = FbkStop.
```

When all feedbacks are false then the drive is between of two positions. In this case no monitoring is started. When two feedbacks are active, then they are not plausible. In this case the static monitoring counter starts. In both cases “all feedbacks are reset” or “not plausible feedbacks” with `Feature2Bit21` =1 all the control outputs are set to false (No command state) else control outputs are hold.

If two feedbacks are available then they are always plausible besides they are both active.

| `NoFbkStop =1` (default) | `NoFbkFwd =1` | `NoFbkRev =1` |
|---|---|---|
| `Fwd = FbkFwd,`<br>`Rev = FbkRev,`<br>`StopCtrl = NOT FbkFwd AND NOT FbkRev` | `Fwd = NOT FbkStop AND NOT FbkRev`<br>`Rev = FbkRev,`<br>`StopCtrl = FbkStop` | `Fwd = FbkFwd,`<br>`Rev = NOT FbkStop AND NOT FbkFwd`<br>`StopCtrl = FbkStop` |

When they are not plausible the control outputs will be hold.

When only one feedback is available the concerned control output is tracked. The other control outputs are hold if the available feedback is false.

| `NoFbkStop =0` | `NoFbkFwd =0` | `NoFbkRev =0` |
|---|---|---|
| `StopCtrl = FbkStop,` `Fwd,Rev` are hold if `FbkStop =0` | `Fwd = FbkFwd,` `StopCtrl,Rev` are hold if `FbkFwd =0` | `Rev = FbkRev,` `StopCtrl,Fwd` are hold if `FbkRev =0` |

When all feedbacks are not available the control outputs will be hold.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and Bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 82) section.

Motor actions you can control in auto mode:

- "Starting in forward" (`FwdAut` = 1)
- "Starting in reverse" (`RevAut` = 1)
- "Stopping" (`StopAut` = 1)

### "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the section Manual and automatic mode for motors, valves and dosers (Page 82).

Motor actions you can control in "manual mode":

- "Starting in forward" (`FwdMan` = 1)
- "Starting in reverse" (`RevMan` = 1)
- "Stopping" (`StopMan` = 1)

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature2.Bit1` =0 and `Feature2.Bit1` =1 is as shown below:

| Output Parameters | **`Feature2.Bit1` =0** | **`Feature2.Bit1` =1** |
|---|---|---|
| FbkFwdOut.ST | 16#80 | 16#60 |
| FbkRevOut.ST | 16#80 | 16#60 |
| RunFwd.ST | 16#80 | 16#60 |
| RunRev.ST | 16#80 | 16#60 |
| Stop.ST | 16#80 | 16#60 |
| AV_Out.ST | 16#80 | 16#60 |

### See also

MotRevL block diagram (Page 1474)
MotRevL I/Os (Page 1463)
MotRevL messaging (Page 1461)
MotRevL error handling (Page 1457)
MotRevL functions (Page 1447)
Description of MotRevL (Page 1440)

## 8.4.3 MotRevL functions

### Functions of MotRevL

The functions for this block are listed below.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" |
| 1 | 1 = Operator can switch to "manual mode" |
| 2 | 1 = Operator can switch to "local mode" |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can stop the motor |
| 5 | 1 = Operator can start the motor in forward |
| 6 | 1 = Operator can start the motor in reverse |
| 7 | 1 = Operator can reset the motor |
| 8 | 1 = Operator can define or change the monitoring time for startup |
| 9 | 1 = Operator can define the monitoring time for the status |
| 10 | 1 = Operator can enable the monitoring time function (Bit 8 - 9) |
| 11 | 1 = Operator can activate the Simulation function |
| 12 | 1 = Operator can activate the Release for maintenance function |
| 13 | 1 = Operator can change the high limit (`AV`) for the alarm |
| 14 | 1 = Operator can change the high limit (`AV`) for the warning |
| 15 | 1 = Operator can change the high limit (`AV`) for the tolerance |
| 16 | 1 = Operator can change the limit (`AV`) for hysteresis |

| Bit | Function |
|---|---|
| 17 | 1 = Operator can lower the limit (AV) for the alarm |
| 18 | 1 = Operator can lower the limit (AV) for the warning |
| 19 | 1 = Operator can lower the limit (AV) for the tolerance |
| 20 | 1 = Operator can activate / deactivate messages via AV_AH_MsgEn |
| 21 | 1 = Operator can activate / deactivate messages via AV_WH_MsgEn |
| 22 | 1 = Operator can activate / deactivate messages via AV_TH_MsgEn |
| 23 | 1 = Operator can activate / deactivate messages via AV_TL_MsgEn |
| 24 | 1 = Operator can activate / deactivate messages via AV_WL_MsgEn |
| 25 | 1 = Operator can activate / deactivate messages via AV_AL_MsgEn |
| 26 | 1 =Operator can change the simulation value SimAV |
| 27 - 29 | Not used |
| 30 | 1 = Operator can define the monitoring time for stopping |
| 31 | 1 = Operator can execute the rapid stop |

The block has the following permissions for the OS1Perm parameter:

| Bit | Function |
|---|---|
| 0 - 15 | reserved |

**Note**

If you interconnect a parameter that is also listed in OS_Perm as a parameter, you have to reset the corresponding OS_Perm bit.

### Restart lock after changing direction of rotation or switching off the motor

Use the input parameter IdleTime to enter a restart lock for changing the direction of rotation or restarting the motor. Use the Feature Bit Enabling direct changeover between forward and reverse (Page 156) to define how the change is to take place. When the "Stop" command is given, the motor goes immediately into "Stop" mode, and IdleTime starts after the feedback (FbkFwd and FbkRev = 0) is given. The motor cannot be started again until the IdleTime has expired.

The IdleTime parameter can be set independently of the MonTiDynamic parameter.

### Limit monitoring of an additional analog value

This block provides the standard function Limit monitoring of an additional analog value (Page 98).

### Limit monitoring with hysteresis

This block provides the standard function Limit monitoring with hysteresis (Page 104). It is performed via the input parameter AV_Hyst.

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Interlocks

This block provides the following interlocks:

**`Feature2.Bit16` = 0:**

- Activation enable
- Interlock without reset ("Interlock")
- Interlock with reset ("Protection")
- Permission for stop ("Permission stop")

**`Feature2.Bit16` = 1:**

- Activation enable forward
- Activation enable reverse
- Interlock forward without reset ("Interlock forward")
- Interlock reverse without reset ("Interlock reverse")
- Interlock forward with reset ("Protection interlock forward")
- Interlock reverse with reset ("Protection interlock reverse")

Refer to the section Interlocks (Page 107) as well as Influence of the signal status on the interlock (Page 112).

### Motor protection function

This block provides the standard function Motor protection function (Page 106).

### Rapid stop for motors

This block provides the standard function Rapid stop for motors (Page 115).

### Disabling interlocks

This block provides the standard function Disabling interlocks (Page 111).

### Resetting the block in case of interlocks or errors

This block provides the standard function Resetting the block in case of interlocks or errors (Page 47).

### External error (`FaultExt`), external control system fault (`CSF`)

This block provides the possibility to pass an external error via the `FaultExt` parameter or an external control system fault via the CSF parameter. See MotRevL error handling (Page 1457).

### Group error

This block provides the standard function Outputting group errors (Page 130).

The following parameters are taken into consideration when forming the group error:

- `CSF`
- `Trip`
- `MonDynErr`
- `MonStaErr`
- `FaultExt`

### Outputting a signal for start readiness

This block provides the standard function Outputting a signal for start readiness (Page 59).

### Forming the group status for interlocks

This block provides the standard function Forming the group status for interlock information (Page 113).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `FbkFwdOut.ST`
- `FbkRevOut.ST`
- `LocalLi.ST`
- `FwdLocal.ST`
- `StopLocal.ST`
- `RevLocal.ST`
- `Trip.ST`
- `AV_Out.ST`
- `FwdChn.ST`
- `RevChn.ST`
- `FwdAut.ST` (only if `Feature2.Bit10` = 1)

- RevAut.ST (only if Feature2.Bit10 = 1)
- StopAut.ST (only if Feature2.Bit10 = 1)

### Considering bad quality of automatic commands or external values

If the Feature2 bit Considering bad quality of automatic commands or external values (Page 200) is set to 1 and one of the following parameters has bad signal status (16#00 or 16#28), the block goes to the neutral position (motor stop) in the "Automatic" mode:

- FwdAut.ST
- RevAut.ST
- StopAut.ST

### Forcing operating modes

This block provides the standard function Forcing operating modes (Page 45).

The following states can be enforced:

- "Start in forward" (FwdForce)
- "Start in reverse" (RevForce)
- "Stop" (StopForce)

**Note**

The Feature Bit Enabling direct changeover between forward and reverse (Page 156) for this block has no function with forced operating states. The motor can always be reversed directly.

### Feedback monitoring

This block provides the standard function Monitoring the feedbacks (Page 104).

### Disabling feedback

This block provides the standard function for Disabling feedback for valve and motors (Page 106).

Feedback monitoring can be deactivated separately for each feedback with NoFbkFwd or NoFbkRev or NoFbkStop as required.

The output signals of feedback are tracked by the control and the other available feedbacks:

| NoFbkFwd | NoFbkRev | NoFbkStop | Comment |
|---|---|---|---|
| 0 | 0 | 1 | **Default:**<br>`FbkFwdOut = FbkFwd;`<br>`FbkRevOut = FbkRev;`<br>`FbkStopOut = StopCtrl AND NOT FbkFwd AND NOT FbkRev;`<br>**Local mode with LocalSetting =2/4:**<br>`FbkFwdOut = FbkFwd;`<br>`FbkRevOut = FbkRev;`<br>`FbkStopOut = NOT FbkFwd AND NOT FbkRev;` |
| 1 | 0 | 1 | `FbkFwdOut = Fwd AND NOT FbkRev;`<br>`FbkRevOut = FbkRev;`<br>`FbkStopOut = StopCtrl AND NOT FbkRev;` |
| 0 | 1 | 1 | `FbkFwdOut = FbkFwd;`<br>`FbkRevOut = Rev AND NOT FbkFwd;`<br>`FbkStopOut = StopCtrl AND NOT FbkFwd;` |
| 1 | 1 | 1 | `FbkFwdOut = Fwd;`<br>`FbkRevOut = Rev;`<br>`FbkStopOut = StopCtrl;` |
| 1 | 0 | 0 | `FbkFwdOut = FbkFwd;`<br>`FbkRevOut = Rev AND NOT FbkFwd AND NOT FbkStop;`<br>`FbkStopOut = FbkStop;`<br>**Local mode with LocalSetting =2/4:**<br>`FbkFwdOut = FbkFwd;`<br>`FbkRevOut = NOT FbkFwd AND NOT FbkStop;`<br>`FbkStopOut = FbkStop;` |

| NoFbkFwd | NoFbkRev | NoFbkStop | Comment |
|---|---|---|---|
| 0 | 1 | 0 | FbkFwdOut = FbkFwd;<br>FbkRevOut = Rev AND NOT FbkFwd AND NOT FbkStop;<br>FbkStopOut = FbkStop;<br>**Local mode with LocalSetting =2/4:**<br>FbkFwdOut = FbkFwd;<br>FbkRevOut = NOT FbkFwd AND NOT FbkStop;<br>FbkStopOut = FbkStop |
| 0 | 0 | 0 | FbkFwdOut = FbkFwd;<br>FbkRevOut = FbkRev;<br>FbkStopOut = FbkStop; |
| 1 | 1 | 0 | FbkFwdOut = Fwd AND NOT FbkStop;<br>FbkRevOut = Rev AND NOT FbkStop;<br>FbkStopOut = FbkStop; |

When all feedbacks are not available the position is hold after a switch to local mode with `LocalSetting` =2/4.
When 2 of 3 feedbacks are not available the position is hold after a switch to local mode with `LocalSetting` =2/4.

### Using StopCtrl with mode "No Cmd"

This block provides the standard function Control with output StopCtrl and state “No Cmd” (Page 214)

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Specify warning times for control functions

This block provides the standard function Specifying warning times for control functions at motors and valves (Page 56).

You can generate warning signals when, for example, motors are started. Warning signals can be generated in the following modes:

- Manual mode (Page 82) (`WarnTiMan` input parameter)
- Automatic mode (Page 82) (`WarnTiAut` input parameter)

You specify the warning times in seconds using the input parameters `WarnTiMan` and `WarnTiAut`. If, for example, a motor is started, then this is displayed at the output parameter with `WarnAct` = 1. The motor then starts after the set warning time has expired and `WarnAct` then goes back to 0.

A corresponding warning is not output if the warning times (`WarnTiMan` or `WarnTiAut`) are specified with a smaller value than the `SampleTime` parameter.

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Additional value (`SimAV`, `SimAV_Li`)

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

### Neutral position

This block provides the standard function Neutral position for motors, valves and controllers (Page 54).

### Output signal as a pulse signal or static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 57).

### Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 230).

### Resynchronization

This block provides the standard function Resynchronization (Page 258).

Resynchronization is switched on over feature bits:

| | |
|---|---|
| Feature2 Bit17 =1 | Resynchronization after CPU startup |
| Feature2 Bit18 =1 | Resynchronization after Simulation or after Out of service |
| Feature2 Bit19 =1 | Resynchronization after control system fault |
| Feature2 Bit20 =1 | Resynchronization with 0->1 trigger at StateTrkOn input |

If there are no restrictions the controls are tracked as follow:

**NoFbkFwd =0, NoFbkRev =0, NoFbkStop =1 (default):**

Fwd := FbkFwd;

Rev := FbkRev;

StopCtrl := NOT FbkFwd AND NOT FbkRev;

In case if there are no plausible feedback positions (FbkFwd =1, FbkRev =1) the controls are not changed.

**NoFbkFwd =0, NoFbkRev =0, NoFbkStop =0:**

Plausible feedback positions (only one feedback signal is active) are tracked as control position.

Fwd := FbkFwd;

Rev := FbkRev;

StopCtrl := FbkStop;

In case if there are no plausible feedback positions (two or three feedback signals are active) the control position depends on Feature2 Bit21.

- Feature2 Bit21 =0 (default): Control position is not changed.
- Feature2 Bit21 =1: Control position becomes "No Cmd" Fwd := 0; Rev := 0; StopCtrl := 0;

**NoFbkFwd =0, NoFbkRev =1, NoFbkStop =0:**

Fwd := FbkFwd AND NOT FbkStop;

Rev := NOT FbkFwd AND NOT FbkStop;

StopCtrl := FbkStop;

In case if there are no plausible feedback positions (FbkFwd =1, FbkStop =1) the controls are not changed.

**NoFbkFwd =1, NoFbkRev =0, NoFbkStop =0:**

Fwd := NOT FbkRev AND NOT FbkStop;

Rev := FbkRev AND NOT FbkStop;

StopCtrl := FbkStop;

In case if there are no plausible feedback positions (FbkRev =1, FbkStop =1) the controls are not changed.

With other combinations of **NoFbkFwd**, **NoFbkRev**, **NoFbkStop** the controls are not changed.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the section Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 3 | Enabling resetting of commands for the control settings (Page 172) |
| 4 | Setting switch or button mode (Page 177) |
| 7 | Enabling direct changeover between forward and reverse (Page 156) |
| 9 | Resetting via input signals in the event of interlocking (Protection) or errors (Page 174) |
| 10 | Exiting local mode (Page 189) |
| 11 | Activating the run time of feedback signals (Page 164) |
| 13 | Separate monitoring time for stopping the motor (Page 180) |
| 14 | Enabling rapid stop via faceplate (Page 179) |
| 17 | Enabling bumpless switchover to automatic mode for valves, motors, and dosers (Page 184) |

| Bit | Function |
|---|---|
| 18 | Activating error state for external process control error CSF (Page 162) |
| 19 | Reset even with locked state (Page 176) |
| 20 | Disable calculation of impulse control in LocalSetting 2 and 4 (Page 200) |
| 21 | Enable bumpless switchover to "Automatic" mode for operator only (Page 183) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 23 | Limits disabled in faceplate (Page 206) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 27 | Interlock display with LocalSetting 2 or 4 (Page 189) |
| 28 | Disabling operating points (Page 156) |
| 29 | Signaling limit violation (Page 181) |
| 30 | Set reset depending on the operating mode or the LiOp parameter (Page 174) |
| 31 | Activating reset of protection / error in manual mode (Page 175) |

In pushbutton operation (Bit 4 = 0) the automatic commands in "automatic" mode are latching; in other words, `FwdAut`, `RevAut`, `StopAut` can be reset to 0 after changing the control. In "manual" and "local" modes, however, the automatic commands are not saved and in the absence of automatic commands the automatic control is tracked.

In switching mode (Bit 4 = 1), control is selected with the static signals `FwdAut`, `RevAut`. If `FwdAut`, `RevAut` inputs are not set, the motor is stopped. Control via `StopAut` is not needed. If the "Activate command reset for control" function (Bit 3 = 1) is also enabled, the inputs `FwdAut`, `RevAut` are reset to the neutral position after evaluation in the block.

### Configurable reactions using the `Feature2` parameter

| Bit | Function |
|---|---|
| 1 | Changes signal status of outputs in OOS (Page 208) |
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 3 | Control priority in the event of invalid or not plausible input commands (Page 186) |
| 4 | Setting switch or button mode for local commands (Page 195) |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) |
| 8 | Forcing commands in the Local mode (Page 198) |
| 9 | Button mode with triggered local control commands (Page 209) |
| 10 | Considering bad quality of automatic commands or external values (Page 200) |
| 11 | Suppress MsgLock and “Out of service” mode for a connected message block (Page 202) |
| 13 | Resetting the Pending Error in Local Mode 1 and 3 (Page 210) |
| 16 | Separate interlock for each direction, position, or speed (Page 180) |
| 17 | Resynchronization after startup (Page 213) |
| 18 | Resynchronization after out of service or after simulation (Page 213) |
| 19 | Resynchronization after CSF (Page 214) |

| Bit | Function |
|---|---|
| 20 | Resynchronization with StateTrkOn input (Page 214) |
| 21 | Control with output StopCtrl and state "No Cmd" (Page 214) |

### Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 239).

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Button labels

This block provides the standard function Labeling of buttons and text (Page 235)

Instance-specific text can be configured for the following parameters:

- `FwdMan`
- `RevMan`
- `StopMan`
- `RapidStp`

### See also

Description of MotRevL (Page 1440)

MotRevL messaging (Page 1461)

MotRevL I/Os (Page 1463)

MotRevL block diagram (Page 1474)

MotRevL modes (Page 1445)

EventTs functions (Page 2046)

## 8.4.4 MotRevL error handling

### Error handling of MotRevL

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Mode switchover error
- Invalid or not plausible input commands
- Control system fault (CSF)

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. |
| 41 | The value for the `LocalSetting` I/O is not within the approved limit of 0 to 4. |
| 42 | `LocalSetting` = 0 or `LocalSetting` = 3 or `LocalSetting` = 4 and `LocalLi` = 1 |
| 51 | `FwdLocal` = 1 and `StopLocal` = 1 and `Feature2.Bit4` = 0<br>`RevLocal` = 1 and `StopLocal` = 1 and `Feature2.Bit4` = 0<br>`FwdLocal` = 1 and `RevLocal` = 1<br>`FwdLocal` = 1 and `Rev` = 1 and `Feature.Bit7` = 1<br>`RevLocal` = 1 and `Fwd` = 1 and `Feature.Bit7` = 1<br>`FwdAut` = 1 and `StopAut` = 1 and `Feature.Bit4` = 0<br>`RevAut` = 1 and `StopAut` = 1 and `Feature.Bit4` = 0<br>`FwdAut` = 1 and `RevAut` = 1<br>`FwdAut` = 1 and `Rev` = 1 and `Feature.Bit7` = 1<br>`RevAut` = 1 and `Fwd` = 1 and `Feature.Bit7` = 1<br>`AutModLi` = 1 and `ManModLi` = 1<br>`FwdForce` = 1 and `StopForce` = 1<br>`RevForce` = 1 and `StopForce` = 1<br>`FwdForce` = 1 and `RevForce` = 1 |
| 52 | `LocalAct` = 1 and `LocalSetting` = 2 or 4 and `SimOn` = 1 |

### Mode switchover error

This error can be output by the block, see the section Error handling (Page 128).

### Invalid or not plausible input commands

This error can be output by the block, see the section Error handling (Page 128).

For the following invalid or not plausible input commands, the control output can be kept or switched to the neutral position. This depends on the function Control priority in the event of invalid or not plausible input commands (Page 186).

| Operating mode | Invalid or not plausible input commands | Control reaction with `Feature2.Bit3` =1 |
|---|---|---|

| | | |
|---|---|---|
| Local: `Localsetting` = 1 or `Localsetting` = 3 | Pushbutton operation for local mode (`Feature2.Bit4` = 0):<br>`FwdLocal` = 1 and `RevLocal` = 1 or<br>`FwdLocal` = 1 and `StopLocal` = 1 or<br>`StopLocal` = 1 and `RevLocal` = 1<br>Switching mode (`Feature2.Bit4` = 1):<br>`FwdLocal` = 1 and `RevLocal` = 1<br>Enabling direct changeover between forward and reverse (`Feature.Bit7` = 1):<br>`FwdLocal` = 1 and `Rev` = 1 or<br>`RevLocal` = 1 and `Fwd` = 1 | Motor is stopped |
| Local: `Localsetting` = 1 or `Localsetting` = 3 and forcing | `FwdForce` = 1 and `RevForce` = 1 or<br>`FwdForce` = 1 and `StopForce` = 1 or<br>`StopForce` = 1 and `RevForce` = 1 | |
| Forcing and no "local mode" | `FwdForce` = 1 and `RevForce` = 1 or<br>`FwdForce` = 1 and `StopForce` = 1 or<br>`StopForce` = 1 and `RevForce` = 1 | |
| "Automatic mode" and no forcing | Pushbutton operation (`Feature.Bit4` = 0):<br>`FwdAut` = 1 and `RevAut` = 1 or<br>`FwdAut` = 1 and `StopAut` = 1 or<br>`StopAut` = 1 and `RevAut` = 1<br>Switching mode (`Feature.Bit4` = 1):<br>`FwdAut` = 1 and `RevAut` = 1<br>Enabling direct changeover between forward and reverse (`Feature.Bit7` = 1):<br>`FwdAut` = 1 and `Rev` = 1 or<br>`RevAut` = 1 and `Fwd` = 1 | |
| "Manual mode" and no forcing | `FwdMan` = 1 and `RevMan` = 1 or<br>`FwdMan` = 1 and `StopMan` = 1 or<br>`StopMan` = 1 and `RevMan` = 1 | |

### Control system fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 128) section for more on this.

### See also

MotRevL block diagram (Page 1474)

MotRevL I/Os (Page 1463)

MotRevL messaging (Page 1461)

Description of MotRevL (Page 1440)

MotRevL modes (Page 1445)

MotRevL functions (Page 1447)

## 8.4.5 MotRevL messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Instance-specific messages

### Process control fault

The following control system error messages can be output:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId1` | SIG 1 | AS process control message - fault | $$BlockComment$$<br>Motor feedback error |
| | SIG 2 | AS process control message - fault | $$BlockComment$$<br>Motor protection triggered |
| | SIG 3 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter `CSF`. If it changes to `CSF` = 1, a process control fault is triggered (`MsgEvId1`, SIG 3).

### Instance-specific messages

You can use up to three instance-specific messages with this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External message 1 |
| | SIG 5 | AS process control message - fault | $$BlockComment$$<br>External message 2 |
| | SIG 6 | AS process control message - fault | $$BlockComment$$<br>External message 3 |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance **MsgEvId1**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |
| 4 | ExtVa104 |
| 5 | ExtVa105 |
| 6 | ExtVa106 |
| 7 | ExtVa107 |
| 8 | ExtVa108 |
| 9 | Reserved |
| 10 | Reserved |

The associated values 4 ... 8 are allocated to the parameters ExtVa104 ... ExtVa108 and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of MotRevL (Page 1440)

MotRevL functions (Page 1447)

MotRevL I/Os (Page 1463)

MotRevL block diagram (Page 1474)

MotRevL error handling (Page 1457)

MotRevL modes (Page 1445)

## 8.4.6 MotRevL I/Os

I/Os of MotRevL

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutModLi* | 1 = "Automatic mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutModOp* | 1 = "Automatic mode" via operator (controlled by ModLiOp = 0) | BOOL | 0 |
| AV | Input additional analog value, to be connected to AV_Out of the AV block | ANY | |
| AV_AH_Lim | Limit high alarm | REAL | 95.0 |
| AV_AL_Lim | Limit low alarm | REAL | 5.0 |
| AV_Hyst | Hysteresis for alarm, warning and tolerance limits | REAL | 1.0 |
| AV_TH_Lim | Limit high tolerance | REAL | 85.0 |
| AV_TL_Lim | Limit low tolerance | REAL | 15.0 |
| AV_WH_Lim | Limit high warning | REAL | 90.0 |
| AV_WL_Lim | Limit low warning | REAL | 10.0 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkTypeData* | This input can not be edit or linked and is for internal usage only! | | |
| BypProt | 1 = Bypassing interlock in "local mode" and in "simulation" | BOOL | 0 |
| CSF | 1 = External error (control system fault) Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| EventTsIn | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| EventTs2In | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTs2In input parameter serves to interconnect the EventTsOut output parameter of the EventTs,Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg3 | Binary input for freely selectable message 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa104 | Associated value 4 for messages (MsgEvID1) | ANY | |
| ExtVa105 | Associated value 5 for messages (MsgEvID1) | ANY | |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| ExtVa108 | Associated value 8 for messages (MsgEvID1) | ANY | |
| FaultExt | 1 = External error<br>Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| FbkFwd | 1= Motor feedback running forward | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| FbkRev | 1= Motor feedback running reverse | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| FbkStop | 1= Motor feedback stopped | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| Feature | I/O for additional functions (Page 1447)<br>1 = Separate monitoring time for stopping the motor | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 13: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0<br>• 0<br>• 0 |
| Feature2 | I/O for additional functions (Page 1447) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| FwdAut* | 1 = Activating forward mode of motor in automatic mode | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FwdChnST | Signal status of output channel Fwd<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| FwdForce | 1 = Forcing activation of motor forward mode | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FwdLocal | 1 = Activation of forward motor operation in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FwdMan* | 1 = Activation of forward motor operation in "manual mode" | BOOL | 0 |
| IdleTime* | Wait time for change of direction or restart in [s] | REAL | 5.0 |
| Intlock | 0 = Interlock / interlock **Motor Forward** is activated Once the interlock condition has cleared, the block can be operated without reset.<br>1 = Interlock / interlock **Motor Forward** is deactivated. | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Intl_En | 1 = Interlock / **Interlock Motor Forward** (interlock, Intlock parameter) is activated | BOOL | 1 |
| IntlRev | 0 = Interlock Motor Reverse is activated. Once the interlock condition has cleared, the block can be operated without reset.<br>1 = Interlock Motor Reverse is deactivated. | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| IntlRevEn | 1 = Interlock Motor Reverse (interlock, IntlRev parameter) is activated | BOOL | 1 |
| LocalLi | 1 = Activate "local mode" via plant signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalOp* | 1 = "Local mode" via operator | BOOL | 0 |
| LocalSetting | Characteristics of Local mode (Page 86) | INT | 0 |
| ManModLi* | 1 = "Manual mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManModOp* | 1 = "Manual mode" via OS operator (controlled by ModLiOp = 0) | BOOL | 1 |
| ModLiOp | Switchover of operating mode between:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Monitor | 1 = Feedback monitoring | BOOL | 1 |
| MonSafePos | 1 = Go to neutral position in the event of monitoring errors  Motors: Stop is the neutral position | BOOL | 1 |
| MonTiDynamic* | Monitoring time for feedback errors  or feedback start error after successful operation in [s] | REAL | 3.0 |
| MonTiDyStop* | Monitoring time for feedback stop errors after successful operation in [s] | REAL | 3.0 |
| MonTiStatic* | Monitoring time for feedback errors without operation in [s] | REAL | 3.0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_RelOp* | 1 = Release for maintenance via OS operator | BOOL | 0 |
| NoFbkFwd | 1= Motor Feedback running forward not present | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| NoFbkRev | 1= Motor Feedback running reverse not present | BOOL | 0 |
| NoFbkStop | 1= Motor feedback stopped not present | BOOL | 1 |
| `Occupied` | 1 = In use by a batch | BOOL | 0 |
| `OosLi` | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `OosOp*` | 1 = "Out of service", via OS operator | BOOL | 0 |
| `OpSt_In` | Input parameter for local operator authorization, connected with the `Out` output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| `OS_Perm` | I/O for operator permissions (Page 1447)<br><br>1 = Operator can set or change the monitoring time for "Control: Start"<br><br>1 = Operator can set or change the monitoring time for "Control: Stop" | STRUCT<br>• Bit 0: BOOL<br>• Bit 8: BOOL<br>• Bit 20: BOOL<br>• Bit 30: BOOL | -<br>• 1<br>• 1<br>• 1<br>• 1 |
| `OS1Perm` | I/O for operator permissions (Page 1447) | STRUCT<br>• Bit 0: BOOL<br>• Bit 15: BOOL | -<br>• 1<br>• 1 |
| `Permit` | 1 = Activation enable / **Activation enable Motor Forward**<br>0 = No activation enable for motor/**Motor Forward** | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| `Perm_En` | 1 = Activation enable / **(activation enable Motor Forward** (enable, Permit parameter)is activated | BOOL | 1 |
| `PermRev` | 1 = Activation enable Motor Reverse<br>0 = No activation enable Motor Reverse | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| PermitSafePos | 1=Motor stop is allowed | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | |
| PermSf_En | 1=Permit stop enabled | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PermRevEn | 1 = Activation enable Motor Reverse(enable, PermRev parameter) is activated | BOOL | 1 |
| Protect | 0 = Protective interlock / **protective interlock Motor Forward** is activated. Once the interlock condition has cleared, the block must be reset<br>1 = Protective interlock / **protective interlock Motor Forward** is deactivated. | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| Prot_En | 1 = Protective interlock / **protective interlock Motor Forward** (protection, Protect parameter) is activated | BOOL | 1 |
| ProtRev | 0 = Protective interlock Motor Reverse is activated. Once the interlock condition has cleared, the block must be reset<br>1 = Protective interlock Motor Reverse is deactivated. | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| ProtRevEN | 1 = Protective interlock Motor Reverse (protection, ProtRev parameter) is activated | BOOL | 1 |
| PulseWidth* | Pulse width of control signal [s] | REAL | 3.0 |
| RapidStp* | Rapid stop for the motor:<br>0 = Motor On<br>1 = Motor Off | BOOL | 0 |
| RevAut* | 1 = Activation of reverse motor operation in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RevChnST | Signal status of output channel Rev<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| RevForce | 1 = Force activation of reverse motor operation | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RevLocal | 1 = Activation of reverse motor operation in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RevMan* | 1 = Activation of reverse motor operation in "manual mode" | BOOL | 0 |
| RstLi* | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | 1 = Reset via operator | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SimAV* | Additional value used for SimOn = 1 | REAL | 0.0 |
| SimAV_Li | Additional analog value that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | - |
| StepNo | Batch step number | DWORD | 16#00000000 |
| StopAut* | 1 = Stop the motor in automatic mode | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopForce | 1 = Force motor stop | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopLocal | 1 = Stopping the motor in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopMan* | 1 = Stopping the motor in "manual mode" | BOOL | 0 |
| Trip | 1 = Motor is in "good" state | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| UserAna1 | Analog auxiliary value 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UA1unit | Unit of measure for analog auxiliary value 1 | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| UserAna2 | Analog auxiliary value 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UA2unit | Unit of measure for analog auxiliary value 2 | INT | 0 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |
| WarnTiAut* | Prewarning of motor start in automatic mode in [s] | REAL | 0.0 |
| StateTrkOn | Initiate tracking of state | BOOL | 0 |
| WarnTiMan* | Prewarning of motor start in manual mode in [s] | REAL | 0.0 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutAct | 1 = "Automatic mode" enabled<br>0 = "Manual mode" is enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutoST | 1 = Worst signal status of automatic commands | BYTE | 16#80 |
| AV_OpScale | Limit for scale in AV bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| AV_Out | Output additional analog value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| AV_Unit | Unit of measure for additional analog value | INT | 0 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| CurrMon | Remaining monitoring time [s] | DINT | 0 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see MotRevL error handling (Page 1457) | INT | -1 |
| FaultExtOut | FaultExt as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| FbkFwdOut | 1= Motor feedback running forward | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkRevOut | 1= Motor feedback running reverse | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkStopOut | 1= Motor feedback stopped | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Fwd | 1 = Control of motor forward | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| GrpErr | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IntlockOut | Intlock as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IntlRevOut | IntlRev as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalAct | 1 = "Local mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LockAct | 1 = Interlock (Intlock, Permit, Protect) or Trip is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManAct | 1 = "Manual mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| MonDynErr | 1 = Feedback error or feedback start error due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MonDynStopErr | 1 = Feedback stop error due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MonStaErr | 1 = Feedback error due to unexpected feedback change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message acknowledgement status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | Alarm error 1 (output ERROR of first ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1Perm | WORD | 16#FFFF |
| OS1PermLog | Display of OS1Perm with settings changed by the block algorithm | WORD | 16#FFFF |
| PermitOut | Permit as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PermitSafePosOut | PermitSafePos as output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| PermRevOut | PermRev as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ProtectOut | Protect as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ProtRevOut | ProtRev as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Fwd | 1 = Pulse signal for starting the motor in forward | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Rev | 1 = Pulse signal for starting the motor in reverse | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| P_Rst | 1= Pulse output for reset<br>The parameter persists for one cycle after a reset. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Stop | 0 = Pulse signal for stopping the motor | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| R_StpAct | 1 = Rapid stop of the motor is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| RdyToReset | 1 = Ready for reset via RstLi input or commands in "local", "automatic", or "manual mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToStart | 1 = Active start readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Rev | 1 = Control of motor: Reverse | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RunFwd | 1 = Motor is running forwards | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RunRev | 1 = Motor is running in reverse | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Status1 | Status word 1 (Page 1440) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1440) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1440) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 1440) | DWORD | 16#00000000 |
| Status5 | Status word 5 (Page 1440) | DWORD | 16#00000000 |
| Stop | 1 = Motor stopped | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopCtrl | Control output to Stop the motor (go to Safe position) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| TripOut | Trip as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| UA1unitOut | UA1 as output | INT | 0 |
| UA2unitOut | UA2 as output | INT | 0 |
| UserAna1Out | UserAna1 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| UserAna2Out | UserAna2 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| ResyncST | Worst Signal Status of inputs for resynchronization | BYTE | 16#80 |
| WarnAct | 1 = Prewarning for motor start active (parameters WarnTiAut and WarnTiMan) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### See also

MotRevL messaging (Page 1461)

MotRevL block diagram (Page 1474)

MotRevL modes (Page 1445)

Error handling (Page 128)

## 8.4.7 MotRevL block diagram

### MotRevL block diagram

A block diagram is not provided for this block.

### See also

MotRevL I/Os (Page 1463)

MotRevL messaging (Page 1461)

MotRevL error handling (Page 1457)

MotRevL functions (Page 1447)

MotRevL modes (Page 1445)

Description of MotRevL (Page 1440)

## 8.4.8 Operator control and monitoring

### 8.4.8.1 MotRevL views

#### Views of the MotRevL block

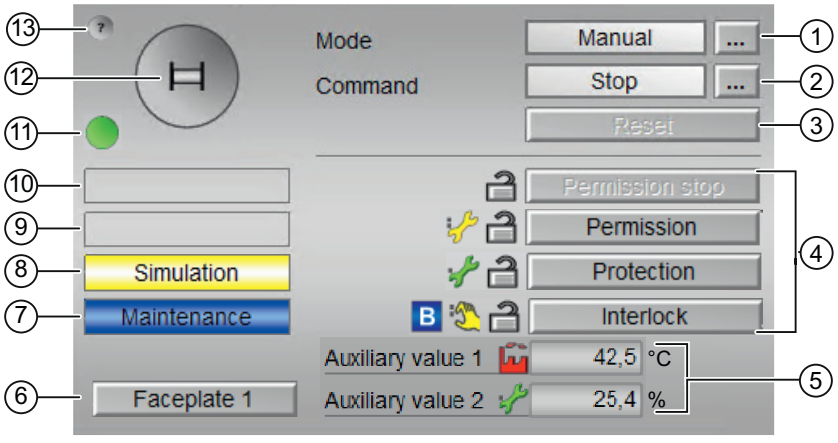
The block MotRevL provides the following views:

- MotRevL standard view (Page 1475)
- Alarm view (Page 336)
- Limit view of motors (Page 328)
- Trend view (Page 341)
- Parameter view for motors and valves (Page 320)
- MotRevL preview (Page 1479)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for MotRevL (Page 1485)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 8.4.8.2 MotRevL standard view

#### MotRevL standard view



### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual and automatic mode for motors, valves and dosers (Page 82)
- Manual and automatic mode for motors, valves and dosers (Page 82)
- Local mode (Page 86)
- Out of service (Page 77)

See also the section Switching operating states and operating modes (Page 289) for information on switching the operating mode.

### (2) Starting and stopping the motor

This area shows you the default operating state for the motor. The following states can be shown and executed here:

- "Start→ "
- "Start ←"
- "Stop"
- "Rapid stop"

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

If the text is configured for these commands, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (3) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in the section Resetting the block in case of interlocks or errors (Page 47).

### (4) Operating range for the interlock functions of the block

This display is only visible when the corresponding block input is connected.

You can use this button to control the interlock functions of the block. You can find additional information on this in the section Interlocking functions (Page 107).

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 113)), for example:



- Signal status (see Forming and outputting the signal status for technologic blocks (Page 117)), for example:



- Bypass information (see Forming the group status for interlock information (Page 113)):



If the text is configured for the interlocks, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (5) Display of auxiliary values

This display is only visible when the corresponding block input is connected.

You can use this area to display two auxiliary values that have been configured in the engineering system (ES). You can find additional information on this in the section Displaying auxiliary values (Page 239).

### (6) Navigation button for switching to the standard view of any faceplate

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234).

### (7) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this is available in section Release for maintenance (Page 70) Display area for block states.

### (8) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"
- "Delay"

You will find more detailed information on this in the sections Simulating signals (Page 64) and Display of delay times (Page 288).

### (9) Display area for block states

This area provides additional information on the operating state of the block:

- "Motor protection"
- "External error"
- "Status error"
- "Control error"
- "Internal error"
- "Changeover error"
- "Invalid Signal"

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 104) , Error handling (Page 128) (section "Invalid input signals" and "Mode switchover error") and Motor protection function (Page 106).

### (10) Display area for block states

This area provides additional information on the operating state of the block:

- "Forced stop"
- "Forced start →"
- "Forced start ←"
- "Request 0/1": Reset is expected but reset is not possible from faceplate. For more details, refer:Reset of errors during Request 0/1 (Page 382)

You can find additional information on this in the section Forcing operating modes (Page 45).

### (11) Automatic preview

This display is only visible in "manual mode", in "local mode", or with a reset request in "automatic mode", when the current output signals are not identical to the control in "automatic mode".

The display shows what state the motor would assume if you switched from "manual" or "local" mode to "automatic mode", or performed a reset to "automatic mode".

### (12) Status display of the motor

The current status of the motor is graphically displayed here.

You can find more information about this in section Block icon for MotRevL (Page 1485).

### (13) Mini preview for Resynchronization

Displays the mini preview for Resynchronizing the signal state.

The mini preview displays the following states:

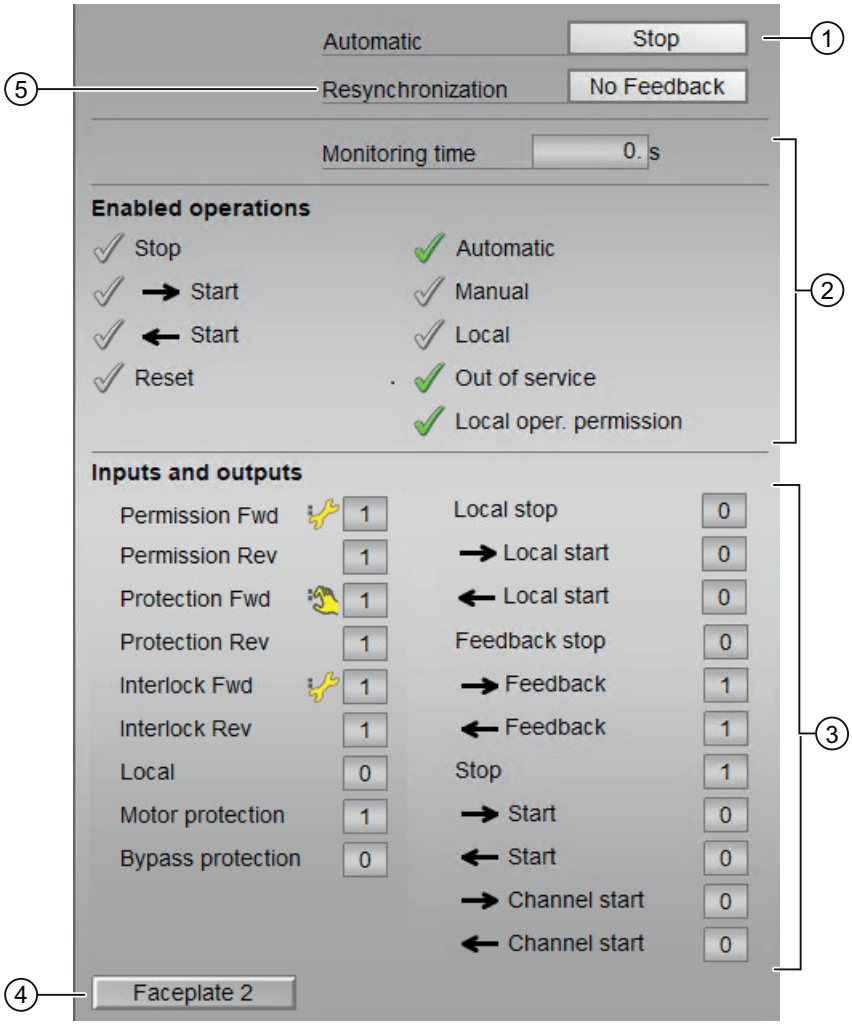
- Fwd
- Rev
- Stop
- No Cmd

**Note**

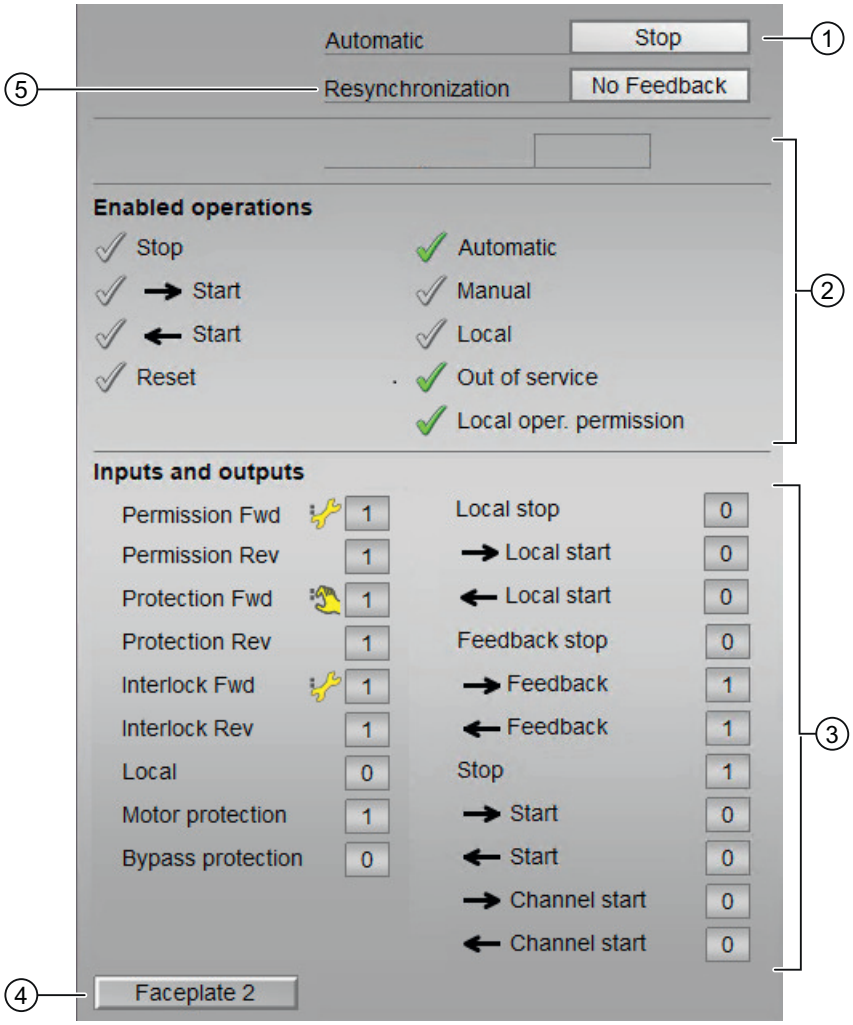
Mini preview for resynch is visible only if resynchronization command is different to control output value.

### 8.4.8.3 MotRevL preview

**Preview of MotRevL**



Display of the current monitoring time is visible.




Display of the current monitoring time is not visible.

### (1) Automatic preview

This area shows you the block status after it has switched from the "Manual" mode or "Local" mode to the "Automatic" mode.

If the block is in "Automatic" mode, the current block state is displayed.

**Note**

- If the valve is closed and interlock locking is active (`SafePos`=0 Close), the automatic preview does not show the current status "Closed", instead it shows the active automatic command "Open".
- If the valve is in the end position and the automatic stop command is active, the automatic preview does not show the current status "Closed", but the active automatic command "Stop".

This area also displays the worst signal status of the following automatic commands:

- `FwdAut`
- `RevAut`
- `StopAut`

### (2) Enable operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "Stop": You can stop the motor.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Start →": You can start the motor.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Start ←": You can start the motor.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Reset": You can reset the motor after interlocks or errors.
- "Automatic": You can switch to "automatic mode".
- "Manual": You can switch to "manual mode".
- "Local": You can switch to "local mode".
- "Out of service": You can switch to "Out of service" operating mode.

- "Local operator permission": Use the ← button to switch to the standard view of the OpStations block. For more information, see section Operator control permissions (Page 285).
- "Monitoring time": Display of the remaining monitoring time.

### (3) Displaying current control signals

This area shows the most important parameters for this block with the current selection:

- "Permission":
  This display is only visible when the corresponding block input is connected.
  - 0 = No OS release for energizing motor
  - 1 = Enable for "starting"/"stopping" from the neutral position
- "Enable" (`Feature2.Bit16` **=0**):
  This display is only visible when the corresponding block input is connected.
  - 0 = No OS release for energizing motor
  - 1 = Enable for "Start" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Enable Fwd" (`Feature2.Bit16` **=1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = No OS release for energizing motor
  - 1 = Enable for "Start forward" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Enable Rev" (`Feature2.Bit16` **=1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = No OS release for energizing motor
  - 1 = Enable for "Start reverse" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection":
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you must reset the block
  - 1 = "Good" state

- "Protection" (`Feature2.Bit16` **=0**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you must reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection Fwd" (`Feature2.Bit16` **=1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you must reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection Rev" (`Feature2.Bit16` **=1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you must reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock":
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state
- "Interlock" (`Feature2.Bit16` **=0**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock Fwd" (`Feature2.Bit16` **=1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).

- "Interlock Rev" (`Feature2.Bit16` **=1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Local": 1 = Control signal for "Local mode" (LocalLi) is active
- "Motor protection": 1 = Motor is in "good" state
- "Interlock deact.":
  - 0 = Bypass disabled
  - 1 = Bypassing interlock in "local mode" and in "simulation"
- "Local stop": 1 = Stopping the motor in "local mode"
- "Local start →": 1 = Starting the motor in "local mode"
- "Local start ←": 1 = Starting the motor in "local mode"
- "Feedback →": 1 = Motor has started and is running
- "Feedback ←": 1 = Motor has started and is running
- "Feedback stop" : 1 = Motor has stopped
- "Stop": 1 = Stop motor
- "Start →": 1 = Start motor
- "Start ←": 1 = Start motor
- "Channel Start →": Signal from the output channel block for "Start"
- "Channel Start ←": Signal from the output channel block for "Start"
- "Permission stop": 1 = Motor can not be stopped in manual (by operator) or automatic (by logic) mode

### (4) Navigation button for switching to the standard view of any faceplate

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

For more information, see section Opening additional faceplates (Page 234).

### (5) Resynchronization

Displays the Resynchronization state and Resynchronization Signal status based on feedback value.

### 8.4.8.4 Block icon for MotRevL

## Block icons for MotRevL

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as control system faults
- Operating modes
- Signal status, release for maintenance
- Displays for bypassing interlocks
- Interlocks
- Memo display
- Motor state display

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | Direction display inverted to symbol 1 |
|  | 4 | Direction display inverted to symbol 2 |
|  | 5 | "M" symbol with small direction display |
|  | 6 | "M" symbol with small direction display |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | |
|  | 6 | |
|  | 7 | |
|  | 8 | |
|  | 9 | |
|  | 10 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

## Motor state display

The following motor states are shown here:

| Icon | Meaning |
|---|---|
|  | Motor started (motor symbol changes) |
|  | The motor is running |
|  | Motor stopped (motor symbol changes) |
|  | Motor idle |
|  | Error at motor (monitoring error, motor protection) |
|  | Motor out of service |
|  | “No Cmd” State |

# 8.5 MotSpdCL - Controllable reversible motor

## 8.5.1 Description of MotSpdCL

### Object name (type + number) and family

Type + number: FB 1854

Family: Drives

### Area of application for MotSpdCL

The block is used for the following applications:

- Control of motors with two directions of rotation and different speeds
- Control of a device feed (optionally configurable via `Feature` bit)

### How it works

The block is used to control reversible motors for different speeds. Various inputs are available for controlling the motor.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

For the MotSpdCL block, the Advanced Process Library contains a template for process tag types as an example with an application scenario for this block.

Example of process tag types:

- Reversible motor with controllable speed (MotorSpeedControlled) (Page 2862)

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

**Note**

At a warm restart with the `Feature` Bit set to 0, the block is switched to manual mode and the setpoint is set to internal and to 0.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

### Status word allocation for `Status1` parameter

For a description of the individual parameters, see the section MotSpdCL I/Os (Page 1518).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | 0 = `ManAct.Value`<br>1 = `AutoAct.Value` |
| 6 | `LocalAct.Value` |
| 7 | 0: Open padlock in the block icon<br>1: Closed padlock in the block icon<br>In case of `Feature2.Bit16 =1` then a closed padlock is displayed if Interlock forward or Interlock reverse is active. |
| 8 | `Fwd.Value` |
| 9 | `SimLiOp.Value` |
| 10 | `Rev.Value` |
| 11 | `MonStaErr.Value` |
| 12 | `MonDynErr.Value` |
| 13 | `BypProt` |
| 14 | Invalid signal status |
| 15 | Mode switchover error |
| 16 | 1 = `Intlock` is active |
| 17 | 1 = `Permit` is active |
| 18 | 1 = `Protect` is active |
| 19 | `Trip` |
| 20 | Display „Forced forward“ |
| 21 | Display „Forced stop“ |
| 22 | Display „Forced reverse“ |
| 23 | "Interlock" button is enabled |
| 24 | Reset request in automatic preview |
| 25 | `WarnAct.Value` or<br>`IdleTime` active |
| 26 | Bypass information for display in block icon |
| 27 | Automatic preview for forward mode |
| 28 | Automatic preview for "stopping" |
| 29 | Automatic preview for reverse mode |
| 30 | "Permission" button is enabled |
| 31 | "Protection" button is enabled |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `MsgLock` |
| 1 | `AV_AH_Act.Value` |
| 2 | `AV_WH_Act.Value` |
| 3 | `AV_TH_Act.Value` |
| 4 | `AV_TL_Act.Value` |
| 5 | `AV_WL_Act.Value` |
| 6 | `AV_AL_Act.Value` |
| 7 | `AV_AH_En` OR `Feature.Bit23` |
| 8 | `AV_WH_En` OR `Feature.Bit23` |
| 9 | `AV_TH_En` OR `Feature.Bit23` |
| 10 | `AV_TL_En` OR `Feature.Bit23` |
| 11 | `AV_WL_En` OR `Feature.Bit23` |
| 12 | `AV_AL_En` OR `Feature.Bit23` |
| 13 | `AV_AH_MsgEn` |
| 14 | `AV_WH_MsgEn` |
| 15 | `AV_TH_MsgEn` |
| 16 | `AV_TL_MsgEn` |
| 17 | `AV_WL_MsgEn` |
| 18 | `AV_AL_MsgEn` |
| 19 | 1 = No impact of input signals on "local mode" with `LocalSetting` = 0, `LocalSetting` = 2, `LocalSetting` = 4 and OOS mode. |
| 20 | Motor is stopped |
| 21 | Motor stops in forward mode |
| 22 | Motor stops in reverse mode |
| 23 | Motor starts in forward mode |
| 24 | Motor runs in forward mode |
| 25 | Motor starts in reverse mode |
| 26 | Motor runs in reverse mode |
| 27 | Error when "stopping" the motor |
| 28 | Error in forward mode of motor |
| 29 | Error in reverse mode of motor |
| 30 | `SP_ExtAct.Value` |
| 31 | Display for interlocks in block icon |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 | 1 = SP ramp active |
| 1 | `RbkWH_Act.Value` |
| 2 | Delay of the `AV_AH_Lim` message |
| 3 | Delay of the `AV_WH_Lim` message |

| Status bit | Parameter |
|---|---|
| 4 | RbkWL_Act.Value |
| 5 | Delay of the AV_TH_Lim message |
| 6 | Delay of the AV_TL_Lim message |
| 7 | RbkWH_En OR Feature.Bit23 |
| 8 | Delay of the AV_WL_Lim message |
| 9 | Delay of the AV_AL_Lim message |
| 10 | RbkWL_En OR Feature.Bit23 |
| 11 | Delay of the RbkWH_Lim message |
| 12 | Delay of the RbkWL_Lim message |
| 13 | Collection of message delays |
| 14 | RbkWH_MsgEn |
| 15 | Current monitoring time is visible |
| 16 | MonDynStopErr.Value |
| 17 | RbkWL_MsgEn |
| 18 | Motor is stopped |
| 19 | 1 = Enable for rapid stop (Feature Bit Enabling rapid stop via faceplate (Page 179)) |
| 20 | SP_RmpModTime |
| 21 | SP_RmpOn |
| 22 | SP_UpRaAct, SP_DnRaAct limits enabled for gradient mode (SP_RateOn = 1) |
| 23 | Command for "rapid stop" of the motor |
| 24 | Command for "starting" → the motor |
| 25 | Command for "starting" ← the motor |
| 26 | Show automatic preview in the standard view |
| 27 | RdyToStart.Value |
| 28 | GrpErr.Value |
| 29 | MS_RelOp |
| 30 | Auxiliary value 1 visible |
| 31 | Auxiliary value 2 visible |

## Status word allocation for Status4 parameter

| Status bit | Parameter |
|---|---|
| 0 | Effective signal 1 of the message block connected via EventTsIn |
| 1 | Effective signal 2 of the message block connected via EventTsIn |
| 2 | Effective signal 3 of the message block connected via EventTsIn |
| 3 | Effective signal 4 of the message block connected via EventTsIn |
| 4 | Effective signal 5 of the message block connected via EventTsIn |
| 5 | Effective signal 6 of the message block connected via EventTsIn |
| 6 | Effective signal 7 of the message block connected via EventTsIn |
| 7 | Effective signal 8 of the message block connected via EventTsIn |
| 8 | AV not connected |
| 9 | Motor protection display (Trip.Status ≠ 16#FF) |

| Status bit | Parameter |
|---|---|
| 10 | 1 = Input parameter FbkFwd is connected |
| 11 | 1 = Input parameter FbkRev is connected |
| 12 | 1 = Setpoint difference high limit violated (ER_AH_Act.Value) |
| 13 | 1 = Setpoint difference low limit violated (ER_AL_Act.Value) |
| 14 | ER_AH_En OR Feature.Bit23 |
| 15 | ER_AL_En OR Feature.Bit23 |
| 16 | 1 = Report setpoint difference high limit violation (ER_AH_MsgEn) |
| 17 | 1 = Report setpoint difference low limit violation (ER_AL_MsgEn) |
| 18 | 1 = Monitoring of setpoint difference "SP - Rbk" activated |
| 19 | DvFdAct.Value Device feed ON |
| 20 | NOT DvFdAct.Value Device feed OFF |
| 21 | Feature.Bit15 Frequency converter with separate device feed (Page 162) |
| 22 | External error generated by FaultExt or external control system fault from CSF with set Feature.Bit18 Activating error state for external process control error CSF (Page 162) |
| 23 | Hidden bypass signal in Permit |
| 24 | Hidden bypass signal in interlock |
| 25 | Hidden bypass signal in Protect |
| 26 | Feature2.Bit2: Separate bypass signal |
| 27 | Feature.Bit16: Setpoint specification with separate display area and custom unit (Page 182) |
| 28 | 1 = Input parameter FwdChnST is interconnected |
| 29 | 1 = Input parameter RevChnST is interconnected |
| 30 | 1 = Input parameter SP_OutChnS is interconnected |
| 31 | Separate monitoring of shutdown of the motor (Feature.Bit13) |

### Status word allocation for Status5 parameter

| Status bit | Parameter |
|---|---|
| 0 - 7 | Effective signal 9..16 of the message block connected via EventTsIn |
| 8 | Delay of the ER_AH_Lim message |
| 9 | Delay of the ER_AL_Lim message |
| 10 | Not used |
| 11 | 1= Suppressed messages |
| 12 | 1 = Input parameter FbkStop is connected |
| 13 - 15 | Not used |
| 16 | 1 = Input parameter SP_LiRmpTime is not interconnected (SP_LiRmpTime.ST = 16#FF) |
| 17 - 26 | Effective signal 1...16 of the message block connected via EventTs2In |
| 27 | Internal error |
| 28-31 | Effective signal 1...16 of the message block connected via EventTs2In |

### Status word allocation for `Status6` parameter

| Status bit | Parameter |
|---|---|
| 0 | Resynchronization preview for Forward |
| 1 | Resynchronization preview for Reverse |
| 2 | Resynchronization preview for Stop |
| 3 | Resynchronization: Display preview |
| 4 | Resynchronization: Display small preview symbol in standard view |
| 5 | Not used |
| 6 | Resynchronization with state "No Cmd" |
| 7 - 11 | Not used |
| 12 | `Feature2 bit 16`: Separate interlock for each direction or position |
| 13 | Hidden bypass signal in Permit Reverse |
| 14 | Hidden bypass signal in Interlock Reverse |
| 15 | Hidden bypass signal in Protect Reverse |
| 16 | "Permission Forward" button is enabled |
| 17 | "Permission Reverse " button is enabled |
| 18 | "Interlock Forward " button is enabled |
| 19 | "Interlock Reverse " button is enabled |
| 20 | "Protect Forward " button is enabled |
| 21 | "Protect Reverse " button is enabled |
| 22 | 1 = Permission Reverse is active |
| 23 | 1 = Intlock Reverse is active |
| 24 | 1 = Protect Reverse is active |
| 25 | 1 = "Permission stop" button is enabled |
| 26 | 1 = Permission stop is active |
| 27 | Hidden bypass signal in Permission stop |
| 28-31 | Not used |

### See also

MotSpdCL block diagram (Page 1535)
MotSpdCL messaging (Page 1516)
MotSpdCL error handling (Page 1512)
MotSpdCL functions (Page 1496)
MotSpdCL modes (Page 1494)

## 8.5.2 MotSpdCL modes

### MotSpdCL operating modes

The block can be operated using the following modes:

- Local mode (Page 86)
- Automatic mode (Page 82)
- Manual mode (Page 82)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

### "Local mode"

You can find general information on "Local mode", switching modes and Bumpless switchover in the Local mode (Page 86) .

Motor actions you can control in local mode:

- "Starting in forward" (`FwdLocal` = 1)
- "Starting in reverse" (`RevLocal` = 1)
- "Stopping" (`StopLocal` = 1)

A motor operated in "local mode" is controlled either by "local" signals or by the feedback signals (input parameters `FbkFwd` = 1, `FbkRev` = 1 or `FbkStop` =1/). Configuration takes place via the input parameter `LocalSetting`.

`LocalSetting` =2,4: The control outputs are tracked to the feedbacks.

All three feedbacks are available and plausible means that always only one feedback is active. The corresponding control output is tracked:

```
Fwd = FbkFwd,
Rev = FbkRev,
StopCtrl = FbkStop.
```

When all feedbacks are false then the drive is between of two positions. In this case no monitoring is started. When two feedbacks are active, then they are not plausible. In this case the static monitoring counter starts. In both cases “all feedbacks are reset” or “not plausible feedbacks” with `Feature2Bit21` =1 all the control outputs are set to false (No command state) else control outputs are hold.

If two feedbacks are available then they are always plausible besides they are both active.

| `NoFbkStop =1` (default) | `NoFbkFwd =1` | `NoFbkRev =1` |
|---|---|---|
| `Fwd = FbkFwd,`<br>`Rev = FbkRev,`<br>`StopCtrl = NOT FbkFwd AND NOT FbkRev` | `Fwd = NOT FbkStop AND NOT FbkRev`<br>`Rev = FbkRev,`<br>`StopCtrl = FbkStop` | `Fwd = FbkFwd,`<br>`Rev = NOT FbkStop AND NOT FbkFwd`<br>`StopCtrl = FbkStop` |

When they are not plausible the control outputs will be hold.

When only one feedback is available the concerned control output is tracked. The other control outputs are hold if the available feedback is false.

| NoFbkStop =0 | NoFbkFwd =0 | NoFbkRev =0 |
|---|---|---|
| StopCtrl = FbkStop,<br>Fwd,Rev are hold if FbkStop =0 | Fwd = FbkFwd,<br>StopCtrl,Rev are hold if FbkFwd =0 | Rev = FbkRev,<br>StopCtrl,Fwd are hold if FbkRev =0 |

When all feedbacks are not available the control outputs will be hold.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and Bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 82) .

Motor actions you can control in auto mode:

- "Starting in forward" (FwdAut = 1)
- "Starting in reverse" (RevAut = 1)
- "Stopping" (StopAut = 1)

### "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the section Manual and automatic mode for motors, valves and dosers (Page 82).

Motor actions you can control in "manual mode":

- "Starting in forward" (FwdMan = 1)
- "Starting in reverse" (RevMan = 1)
- "Stopping" (StopMan = 1)

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with Feature2.Bit1 =0 and Feature2.Bit1 =1 is as shown below:

| Output Parameters | **Feature2.Bit1 =0** | **Feature2.Bit1 =1** |
|---|---|---|
| DvFdAct.ST | 16#80 | 16#60 |
| FbkFwdOut.ST | 16#80 | 16#60 |
| FbkRevOut.ST | 16#80 | 16#60 |
| RunFwd.ST | 16#80 | 16#60 |
| RunRev.ST | 16#80 | 16#60 |
| Stop.ST | 16#80 | 16#60 |
| Starting.ST | 16#80 | 16#60 |

| Output Parameters | `Feature2.Bit1` =0 | `Feature2.Bit1` =1 |
|---|---|---|
| Stoping.ST | 16#80 | 16#60 |
| SP_Out.ST | 16#80 | 16#60 |
| SP_Out2.ST | 16#80 | 16#60 |
| SP_ExtAct.ST | 16#80 | 16#60 |
| RbkOut.ST | 16#80 | 16#60 |
| RbkWH_Act.ST | 16#XX | 16#60 |
| RbkWL_Act.ST | 16#XX | 16#60 |
| ER.ST | 16#80 | 16#60 |
| ER_AH_Act.ST | 16#XX | 16#60 |
| ER_AL_Act.ST | 16#XX | 16#60 |
| AV_Out.ST | 16#80 | 16#60 |

**Note**

In case of external setpoint value specification, the block switches to internal setpoint value specification.

**See also**

MotSpdCL block diagram (Page 1535)

MotSpdCL I/Os (Page 1518)

MotSpdCL messaging (Page 1516)

MotSpdCL error handling (Page 1512)

MotSpdCL functions (Page 1496)

Description of MotSpdCL (Page 1488)

## 8.5.3 MotSpdCL functions

### Functions of MotSpdCL

The functions for this block are listed below.

### Alarm delays with two time values per limit pair

This block has the standard alarm delay function for Two time values per limit pair (Page 227) limit monitoring of the feedback.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the OS_Perm parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" |
| 1 | 1 = Operator can switch to "manual mode" |
| 2 | 1 = Operator can switch to "local mode" |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can stop the motor |
| 5 | 1 = Operator can start the motor in forward |
| 6 | 1 = Operator can start the motor in reverse |
| 7 | 1 = Operator can reset the motor |
| 8 | 1 = Operator can define or change the monitoring time for startup |
| 9 | 1 = Operator can define the monitoring time for the status |
| 10 | 1 = Operator can activate the monitoring time |
| 11 | 1 = Operator can activate the Simulation function |
| 12 | 1 = Operator can activate the Release for maintenance function |
| 13 | 1 = Operator can change the limit (AV) for high alarm |
| 14 | 1 = Operator can change the limit (AV) for high warning |
| 15 | 1 = Operator can change the limit (AV) for high tolerance |
| 16 | 1 = Operator can change the limit (AV) for hysteresis |
| 17 | 1 = Operator can change the limit (AV) for low alarm |
| 18 | 1 = Operator can change the limit (AV) for low warning |
| 19 | 1 = Operator can change the limit (AV) for low tolerance |
| 20 | 1 = Operator can enable the bumpless switchover from external to internal SP_TrkExt |
| 21 | 1 = Operator can change the internal setpoint SP_Int |
| 22 | 1 = Operator can switch the setpoint to "external" SP_ExtOp |
| 23 | 1 = Operator can switch the setpoint to "internal" SP_IntOp |
| 24 | 1 = Operator can enable the setpoint's gradient limitation function SP_RateOn |
| 25 | 1 = Operator can change the setpoint's high limit for the ramp SP_UpRaLim |
| 26 | 1 = Operator can change the setpoint's low limit for the ramp SP_DnRaLim |
| 27 | 1 = Operator can enable the setpoint ramp function SP_RmpOn |
| 28 | 1 = Operator can switch between the time value or the value for the ramp SP_RmpModTime |
| 29 | 1 = Operator can change the ramp time SP_RmpTime |
| 30 | 1 = Operator can change the target setpoint SP_RmpTarget for the setpoint ramp |
| 31 | 1 = Operator can enable rapid stop |

The block has the following permissions for the OS1Perm parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can change the limit (Rbk) for high warning |
| 2 | Not used |

| Bit | Function |
|---|---|
| 3 | 1 = Operator can change the limit (Rbk) for hysteresis |
| 4 | Not used |
| 5 | 1 = Operator can change the limit (Rbk) for low warning |
| 6 | 1 = Operator can change the limit (setpoint difference) ER_AH_Lim for the high alarm |
| 7 | 1 = Operator can change the hysteresis (setpoint difference) ER_Hyst |
| 8 | 1 = Operator can change the limit (setpoint difference) ER_AL_Lim for the low alarm |
| 9 | 1 = Operator can change the simulation value SimRbk |
| 10 | 1 = Operator can activate the device infeed DvFdOnMan |
| 11 | 1 = Operator can deactivate the device infeed DvFdOffMan |
| 12 | 1 =Operator can change the simulation value SimAV |
| 13 | 1 = Operator can change the derivative gain parameter ER_AH_DFac |
| 14 | 1 = Operator can change the derivative gain parameter ER_AL_DFac |
| 15 - 19 | Not used |
| 20 | 1 = Operator can activate / deactivate messages via AV_AH_MsgEn |
| 21 | 1 = Operator can activate / deactivate messages via AV_WH_MsgEn |
| 22 | 1 = Operator can activate / deactivate messages via AV_TH_MsgEn |
| 23 | 1 = Operator can activate / deactivate messages via AV_TL_MsgEn |
| 24 | 1 = Operator can activate / deactivate messages via AV_WL_MsgEn |
| 25 | 1 = Operator can activate / deactivate messages via AV_AL_MsgEn |
| 26 | 1 = Operator can activate / deactivate messages via RbkWH_MsgEn |
| 27 | 1 = Operator can activate / deactivate messages via RbkWL_MsgEn |
| 28 | 1 = Operator can activate / deactivate messages via ER_AH_MsgEn |
| 29 | 1 = Operator can activate / deactivate messages via ER_AL_MsgEn |
| 30 | 1 = Operator can define the monitoring time for stopping |
| 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in OS_Perm as a parameter, you have to reset the corresponding OS_Perm bit.

### Restart lock after changing direction of rotation or switching off the motor

Use the input parameter IdleTime to enter a restart lock for changing the direction of rotation or restarting the motor. Use the Feature Bit Enabling direct changeover between forward and reverse (Page 156) to define how the change is to take place. When the "Stop" command is given, the motor goes immediately into Stop mode and IdleTime starts after the feedback (FbkFwd and FbkRev = 0) is given. The motor cannot be started again until the IdleTime has expired.

The IdleTime parameter can be set independently of the MonTiDynamic parameter.

### Limit monitoring of an additional analog value

This block provides the standard function Limit monitoring of an additional analog value (Page 98).

### Limit monitoring of the feedback

The block provides the standard function Limit monitoring of the feedback (Page 101). This limit monitoring is only active when the motor has started.

### Limit monitoring with hysteresis

This block provides the standard function Limit monitoring with hysteresis (Page 104). It is performed via the input parameter `AV_Hyst`.

### Providing limits at the output

For further connections to other blocks the following input parameters will also be displayed on the output:

`ER_AH_LimOut` := `ER_AH_Lim`;

`ER_AL_LimOut` := `ER_AL_Lim`;

### Group display SumMsgAct for limit monitoring, CSF and ExtMsgx

The block provides the standard function Group display for limit monitoring, CSF and ExtMsgx (Page 92).

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### External/internal setpoint specification

The block provides the standard function Setpoint specification - internal/external (Page 136).

### Setpoint adaption in motor stop and starting

With `Feature2.Bit7` = 1; Define the setpoint after stop and start of the motor (Page 198), the setpoint can be set to a stop value and after start of the motor, the setpoint can be set to a minimal value (`SP_LoLim`).

### Setpoint limitation

Limit the setpoint using the parameters:

- `SP_HiLim` (top)
- `SP_LoLim` (bottom)

Limit violations are displayed at the SP_HiAct and SP_LoAct output parameters with a 1.

With Feature2.Bit7 = 1; Define the setpoint after stop and start of the motor (Page 198), the lower limit value of the setpoint SP_LoLim will be limited to SP_Off and in case of violation, SP_LoLim will be written back to SP_Off.

### Gradient limit of the setpoint

The block provides the standard function Gradient limit of the setpoint (Page 132).

### Using setpoint ramp

The block provides the standard function Using setpoint ramp (Page 131).

### Gear reduction or setpoint adaptation with factor

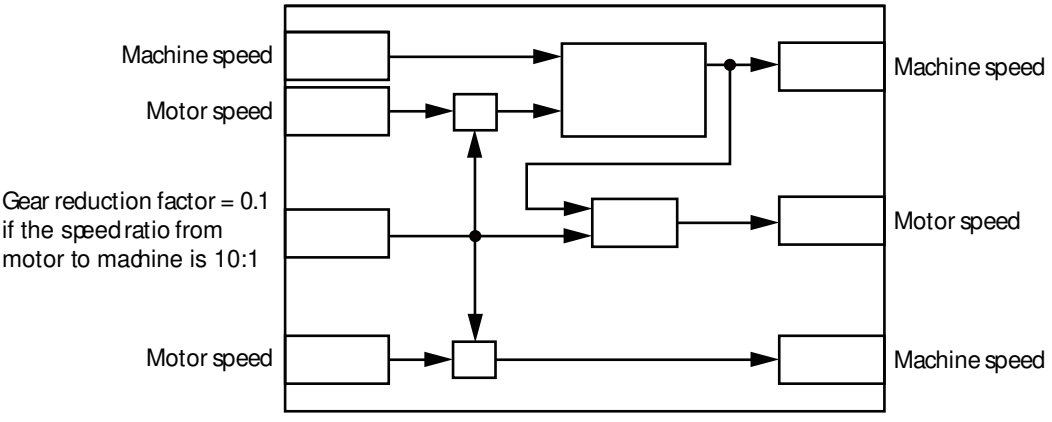
With the parameter FactGR, a factor can be entered for gear reduction. It affects the outputs SP_Out, SP_Out2, and RbkOut:

- SP_Out = FactGR * SP_Ext if the setpoint is external (SP_ExtAct = 1)
- SP_Out2 parameter:
  - If FactGR <> 0.0 then SP_Out2 = SP_Out / FactGR
  - If FactGR = 0.0 then SP_Out2 = SP_Out

As a result, the readback value will also be adjusted:

RbkOut = FactGR * Rbk

Below is an example for the gear reduction from motor speed to machine speed in the ratio 10:1.




The factor FactGR can also be used for adapting the range in percentage.

### Formation of the difference between setpoint and position feedback

The block always forms the difference between setpoint and position feedback:

ER.Value = SP_Out - RbkOut.Value.

### Limit monitoring of the setpoint difference

The block provides the standard function Limit monitoring of setpoint, manipulated variable and control deviation (Page 102).

To transmit the messages, you need to enable `Feature.Bit5` Alarm setpoint difference (Page 182).

When the function activated, the message configuration should be adapted as follows:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId1` | SIG 7 | Alarm - high | $$BlockComment$$<br>`ER` - high alarm limit `SP - Rbk` violated |
| | SIG 8 | Alarm - low | $$BlockComment$$<br>`ER` - low alarm limit `SP - Rbk` violated |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Interlocks

This block provides the following interlocks:

`Feature2` bit 16 = 0:

- Activation enable
- Interlock without reset ("Interlock")
- Interlock with reset ("Protection")
- Permission for stop ("Permission stop")

`Feature2` bit 16 = 1:

- Activation enable forward
- Activation enable reverse
- Interlock forward without reset ("Interlock forward ")
- Interlock reverse without reset ("Interlock reverse ")
- Interlock forward with reset ("Protection interlock forward ")
- Interlock reverse with reset ("Protection interlock reverse ")

Refer to the section Interlocks (Page 107) as well as Influence of the signal status on the interlock (Page 112).

### Motor protection function

This block provides the standard function Motor protection function (Page 106).

### Rapid stop for motors

This block provides the standard function Rapid stop for motors (Page 115).

### Disabling interlocks

This block provides the standard function Disabling interlocks (Page 111).

### Resetting the block in case of interlocks or errors

This block provides the standard function Resetting the block in case of interlocks or errors (Page 47).

### External error (`FaultExt`), external control system fault (`CSF`)

This block provides the possibility to pass an external error via the `FaultExt` parameter or an external control system fault via the CSF parameter. See MotSpdCL error handling (Page 1512).

### Group error

This block provides the standard function Outputting group errors (Page 130).

The following parameters are taken into consideration when forming the group error:

- `CSF`
- `Trip`
- `MonDynErr`
- `MonStaErr`
- `FaultExt`

### Outputting a signal for start readiness

This block provides the standard function Outputting a signal for start readiness (Page 59).

### Forming the group status for interlocks

This block provides the standard function Forming the group status for interlock information (Page 113).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `FbkFwdOut.ST`
- `FbkRevOut.ST`

- LocalLi.ST
- FwdLocal.ST
- StopLocal.ST
- RevLocal.ST
- Trip.ST
- AV_Out.ST
- RbkOut.ST
- SP_Out.ST
- FwdChn.ST
- RevChn.ST
- SP_OutChn.ST
- FwdAut.ST (only if Feature2.Bit10 = 1)
- RevAut.ST (only if Feature2.Bit10 = 1)
- StopAut.ST (only if Feature2.Bit10 = 1)
- SP_Ext.ST (only if Feature2.Bit10 = 1)

The signal status of the SP_Out output parameter is always equivalent to the signal status of input parameter SP_Ext or SP_Int, depending on how the setpoint is specified. If the internal setpoint SP_Int is used, the signal status is always output as 16#80.

In case of Feature.Bit10 = 1, SP_Ext.ST influences ST_Worst independent of setpoint specification.

### Considering bad quality of automatic commands or external values

If the Feature2 bit Considering bad quality of automatic commands or external values (Page 200) is set to 1 and one of the following parameters has bad signal status (16#00 or 16#28), the block goes to the neutral position (motor stop) in the "Automatic" mode:

- FwdAut.ST
- RevAut.ST
- StopAut.ST
- SP_Ext.ST

### Forcing operating modes

This block provides the standard function Forcing operating modes (Page 45).

The following states can be enforced:

- "Start in forward" (`FwdForce`)
- "Start in reverse" (`RevForce`)
- "Stop" (`StopForce`)

**Note**

The `Feature` bit Enabling direct changeover between forward and reverse (Page 156) for this block has no function with forced operating states. The motor can always be reversed directly.

## Feedback monitoring

This block provides the standard function Monitoring the feedbacks (Page 104).

## Disabling feedback

This block provides the standard function for Disabling feedback for valve and motors (Page 106).

Feedback monitoring can be deactivated separately for each feedback with `NoFbkFwd` or `NoFbkRev` or `NoFbkStop` as required.

The output signals of feedback are tracked by the control and the other available feedbacks:

| NoFbkFwd | NoFbkRev | NoFbkStop | Comment |
|---|---|---|---|
| 0 | 0 | 1 | **Default:**<br>`FbkFwdOut = FbkFwd;`<br>`FbkRevOut = FbkRev;`<br>`FbkStopOut = StopCtrl AND NOT FbkFwd AND NOT FbkRev;`<br>**Local mode with LocalSetting =2/4:**<br>`FbkFwdOut = FbkFwd;`<br>`FbkRevOut = FbkRev;`<br>`FbkStopOut = NOT FbkFwd AND NOT FbkRev;` |
| 1 | 0 | 1 | `FbkFwdOut = Fwd AND NOT FbkRev;`<br>`FbkRevOut = FbkRev;`<br>`FbkStopOut = StopCtrl AND NOT FbkRev;` |

| NoFbkFwd | NoFbkRev | NoFbkStop | Comment |
|---|---|---|---|
| 0 | 1 | 1 | `FbkFwdOut = FbkFwd;`<br>`FbkRevOut = Rev AND NOT FbkFwd;`<br>`FbkStopOut = StopCtrl AND NOT FbkFwd;` |
| 1 | 1 | 1 | `FbkFwdOut = Fwd;`<br>`FbkRevOut = Rev;`<br>`FbkStopOut = StopCtrl;` |
| 1 | 0 | 0 | `FbkFwdOut = Fwd AND NOT FbkRev AND NOT FbkStop;`<br>`FbkRevOut = FbkRev;`<br>`FbkStopOut = FbkStop;`<br>**Local mode with LocalSetting =2/4:**<br>`FbkFwdOut = NOT FbkRev AND NOT FbkStop;`<br>`FbkRevOut = FbkRev;`<br>`FbkStopOut = FbkStop;` |
| 0 | 1 | 0 | `FbkFwdOut = FbkFwd;`<br>`FbkRevOut = Rev AND NOT FbkFwd AND NOT FbkStop;`<br>`FbkStopOut = FbkStop;`<br>**Local mode with LocalSetting =2/4:**<br>`FbkFwdOut = FbkFwd;`<br>`FbkRevOut = NOT FbkFwd AND NOT FbkStop;`<br>`FbkStopOut = FbkStop` |
| 0 | 0 | 0 | `FbkFwdOut = FbkFwd;`<br>`FbkRevOut = FbkRev;`<br>`FbkStopOut = FbkStop;` |
| 1 | 1 | 0 | `FbkFwdOut = Fwd AND NOT FbkStop;`<br>`FbkRevOut = Rev AND NOT FbkStop;`<br>`FbkStopOut = FbkStop;` |

When all feedbacks are not available the position is hold after a switch to local mode with `LocalSetting` =2/4.
When 2 of 3 feedbacks are not available the position is hold after a switch to local mode with `LocalSetting` =2/4.

### Using StopCtrl with mode "No Cmd"

This block provides the standard function Control with output StopCtrl and state “No Cmd” (Page 214)

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Specify warning times for control functions

This block provides the standard function Specifying warning times for control functions at motors and valves (Page 56).

You can generate warning signals when, for example, motors are started. Warning signals can be generated in the following modes:

- Manual mode (Page 82) (`WarnTiMan` input parameter)Automatic mode (Page 82) (`WarnTiAut` input parameter)

You specify the warning times in seconds using the input parameters `WarnTiMan` and `WarnTiAut`. If, for example, a motor is started, then this is displayed at the output parameter with `WarnAct` = `1`. The motor then starts after the set warning time has expired and `WarnAct` then goes back to 0 .

A corresponding warning is not output if the warning times (`WarnTiMan` or `WarnTiAut`) are specified with a smaller value than the `SampleTime` parameter.

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Additional value (`SimAV`, `SimAV_Li`)
- Position feedback (`SimRbk`, `SimRbkLi`)

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

### Specifying the display and operator input area for process values and setpoints during operation

This block features the standard function Display and control fields for process values and setpoints (Page 233).

### Neutral position

This block provides the standard function Neutral position for motors, valves and controllers (Page 54).

### Output signal as a pulse signal or static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 57).

### Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 230).

### Delay alarm for control deviation at setpoint step changes

The block provides the standard function Delay alarm for control deviation at setpoint step changes (Page 216).

### Disabling of limits

The block provides the standard function Limits disabled in faceplate (Page 206).

### Resynchronization

This block provides the standard function Resynchronization (Page 258).

Resynchronization is switched on over feature bits:

| | |
|---|---|
| Feature2 Bit17 =1 | Resynchronization after CPU startup |
| Feature2 Bit18 =1 | Resynchronization after Simulation or after Out of service |
| Feature2 Bit19 =1 | Resynchronization after control system fault |
| Feature2 Bit20 =1 | Resynchronization with 0->1 trigger at StateTrkOn input |

If there are no restrictions the controls are tracked as follow:

**NoFbkFwd =0, NoFbkRev =0, NoFbkStop =1 (default):**

Fwd := FbkFwd;

Rev := FbkRev;

StopCtrl := NOT FbkFwd AND NOT FbkRev;

In case if there are no plausible feedback positions (FbkFwd =1, FbkRev =1) the controls are not changed.

**NoFbkFwd =0, NoFbkRev =0, NoFbkStop =0:**

Plausible feedback positions (only one feedback signal is active) are tracked as control position.

Fwd := FbkFwd;

Rev := FbkRev;

StopCtrl := FbkStop;

In case if there are no plausible feedback positions (two or three feedback signals are active) the control position depends on Feature2 Bit21.

- Feature2 Bit21 =0 (default): Control position is not changed.
- Feature2 Bit21 =1: Control position becomes "No Cmd" Fwd := 0; Rev := 0; StopCtrl := 0;

**NoFbkFwd =0, NoFbkRev =1, NoFbkStop =0:**

Fwd := FbkFwd AND NOT FbkStop;

Rev := NOT FbkFwd AND NOT FbkStop;

StopCtrl := FbkStop;

In case if there are no plausible feedback positions (FbkFwd =1, FbkStop =1) the controls are not changed.

**NoFbkFwd =1, NoFbkRev =0, NoFbkStop =0:**

Fwd := NOT FbkRev AND NOT FbkStop;

Rev := FbkRev AND NOT FbkStop;

StopCtrl := FbkStop;

In case if there are no plausible feedback positions (FbkRev =1, FbkStop =1) the controls are not changed.

Analog Readback value:

SP_Out := Rbk; SP_Out is tracked and limited to SP_HiLim and SP_LoLim.

With other combinations of **NoFbkFwd**, **NoFbkRev**, **NoFbkStop** the controls are not changed.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 3 | Enabling resetting of commands for the control settings (Page 172) |
| 4 | Setting switch or button mode (Page 177) |
| 5 | Alarm setpoint difference (Page 182) |
| 6 | Ramp rate calculation (Page 191) |
| 7 | Enabling direct changeover between forward and reverse (Page 156) |
| 8 | Inverter enable (Page 201) |
| 9 | Resetting via input signals in the event of interlocking (Protection) or errors (Page 174) |
| 10 | Exiting local mode (Page 189) |
| 11 | Activating the run time of feedback signals (Page 164) |
| 12 | Gradient limitation with time duration (Page 197) |
| 13 | Separate monitoring time for stopping the motor (Page 180) |

| Bit | Function |
|---|---|
| 14 | Enabling rapid stop via faceplate (Page 179) |
| 15 | Frequency converter with separate device feed (Page 162) |
| 16 | Setpoint specification with separate display area and custom unit (Page 182) |
| 17 | Enabling bumpless switchover to automatic mode for valves, motors, and dosers (Page 184) |
| 18 | Activating error state for external process control error CSF (Page 162) |
| 19 | Reset even with locked state (Page 176) |
| 20 | Disable calculation of impulse control in LocalSetting 2 and 4 (Page 200) |
| 21 | Enable bumpless switchover to "Automatic" mode for operator only (Page 183) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 23 | Limits disabled in faceplate (Page 206) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 27 | Interlock display with LocalSetting 2 or 4 (Page 189) |
| 28 | Disabling operating points (Page 156) |
| 29 | Signaling limit violation (Page 181) |
| 30 | Set reset depending on the operating mode or the LiOp parameter (Page 174) |
| 31 | Activating reset of protection / error in manual mode (Page 175) |

In pushbutton operation (Bit 4 = 0) , the automatic commands in "automatic" mode are latching; in other words, `FwdAut`, `RevAut`, `StopAut` can be reset to 0 after changing the control. In "manual" and "local" modes, however, the automatic commands are not saved and in the absence of automatic commands the automatic control is tracked.

In switching mode (Bit 4 = 1), control is selected with the static signals `FwdAut`, `RevAut`. If `FwdAut`, `RevAut` inputs are not set, the motor is stopped. Control via `StopAut` is not needed. If the "Activate command reset for control" function (Bit 3 = 1) is also enabled, the inputs `FwdAut`, `RevAut` are reset to the neutral position after evaluation in the block.

### Configurable reactions using the `Feature2` parameter

| Bit | Function |
|---|---|
| 1 | Changes signal status of outputs in OOS (Page 208) |
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 3 | Control priority in the event of invalid or not plausible input commands (Page 186) |
| 4 | Setting switch or button mode for local commands (Page 195) |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) |
| 6 | Operator can change the setpoint via faceplate also in the "Local" mode (Page 197) |
| 7 | Define the setpoint after stop and start of the motor (Page 198) |
| 8 | Forcing commands in the Local mode (Page 198) |
| 9 | Button mode with triggered local control commands (Page 209) |
| 10 | Considering bad quality of automatic commands or external values (Page 200) |
| 11 | Suppress MsgLock and “Out of service” mode for a connected message block (Page 202) |

| Bit | Function |
|---|---|
| 12 | Disable limit act output when limit value message is disabled (Page 209) |
| 13 | Resetting the Pending Error in Local Mode 1 and 3 (Page 210) |
| 16 | Separate interlock for each direction, position, or speed (Page 180) |
| 17 | Resynchronization after startup (Page 213) |
| 18 | Resynchronization after out of service or after simulation (Page 213) |
| 19 | Resynchronization after CSF (Page 214) |
| 20 | Resynchronization with StateTrkOn input (Page 214) |
| 21 | Control with output StopCtrl and state “No Cmd” (Page 214) |

### Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 239).

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Button labels

This block provides the standard function Labeling of buttons and text (Page 235)

Instance-specific text can be configured for the following parameters:

- `FwdMan`
- `RevMan`
- `StopMan`
- `RapidStp`

### Controlling a device infeed or an inverter enable

The function is activated via `Feature.Bit15` Frequency converter with separate device feed (Page 162) and specified via `Feature.Bit8` Inverter enable (Page 201).

When the function is activated, the motor can be started in two steps:

- Switch on device infeed or inverter enable first, then start the motor

When the "device infeed" function is activated, the motor can also be started directly:
The device infeed is also activated when starting in forward or reverse. The frequency converter automatically determines the correct sequence. The activation of the device infeed for starting the motor can be performed in the automatic, manual and local operating modes with local setting 1 or 3, or by forcing.

With local setting 2 or 4, the controls are tracked based on feedback signals regardless of the status of the device infeed or the inverter enable.

With motors without device infeed or inverter enable (`Feature.Bit15` = 0 default), the `DvFdAct.Value` control is always zero.

Separate activation/deactivation of the device infeed or the inverter enable is performed in manual mode using `DvFdOnMan` and `DvFdOffMan` via the standard view of the faceplate. In automatic mode, activation/deactivation is performed via the input signals `DvFdOnAut` or `DvFdOffAut`. The `Feature.Bit3` (activate reset of commands for control) and 4 (set switch or pushbutton operation) are also based on the automatic inputs `DvFdOnAut` and `DvFdOffAut`.

The following table shows the effects on MotSpdCL block when parameterizing device infeed and inverter enable:

| **MotSpdCL specification** | **Effect on MotSpdCL block when parameterizing device infeed**<br>**`Feature.Bit8` = 0** | **Effect on MotSpdCL block when parameterizing inverter enable**<br>**`Feature.Bit8` = 1** |
|---|---|---|
| Motor start - dependent (`Fwd` = 1 or `Rev` = 1) | Device infeed is activated (`DvFdAct` = 1) | Inverter enable remains unchanged |
| Motor stop (`Fwd` = 0 and `Rev` = 0) | Device infeed remains unchanged | Inverter enable remains unchanged |
| Activate device infeed or inverter enable (`DvFdAct` = 1) | Motor control remains unchanged | Motor control remains unchanged |
| Deactivate device infeed or inverter enable (`DvFdAct` = 0) | Motor control is reset (`Fwd` = 0, `Rev` = 0) | Motor control remains unchanged |
| Monitoring the feedback signals (`FbkFwd`, `FbkRev`) | Feedback signals are monitored for motor controls (`Fwd`, `Rev`) | Feedback signals are monitored for motor controls and the inverter enable:<br>(`Fwd` and `DvFdAct`), (`Rev` and `DvFdAct`) |
| Monitoring fault, external fault, motor protection, or interlock | Motor control is reset (`Fwd` = 0, `Rev` = 0)<br>Device infeed remains unchanged | Motor control is reset (`Fwd` = 0, `Rev` = 0)<br>Inverter enable remains unchanged |
| Limit value monitoring of the feedback<br>Limit value monitoring of the setpoint difference | Only active when:<br>`Fwd` = 1 or `Rev` = 1 | Only active if the inverter enable is also activated:<br>`Fwd` = 1 or `Rev` = 1;<br>And `DvFdAct` = 1 |

Commands for starting the motor have higher priority than commands for a device infeed or the inverter enable.

**⚠ DANGER**

**Separate switching on/off of the device infeed**

With switch mode (`Feature.Bit4` = 1) and deactivated control reset (`Feature.Bit3` = 0), the `DvFdOnAut` input is also set for controlling the device infeed or the inverter enable when starting in forward or in reverse in automatic mode.

### See also

MotSpdCL block diagram (Page 1535)

MotSpdCL I/Os (Page 1518)

MotSpdCL messaging (Page 1516)

MotSpdCL modes (Page 1494)

Description of MotSpdCL (Page 1488)

EventTs functions (Page 2046)

## 8.5.4 MotSpdCL error handling

### Error handling of MotSpdCL

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Mode switchover error
- Control system fault (CSF)
- Invalid or not plausible input commands

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. |
| 41 | The value for the `LocalSetting` I/O is not within the approved limit of 0 to 4. |
| 42 | `LocalSetting` = 0 or `LocalSetting` = 3 or `LocalSetting` = 4 and `LocalLi` = 1 |

| Error number | Meaning of the error number |
|---|---|
| 51 | FwdLocal = 1 and StopLocal = 1 and Feature2.Bit4 = 0<br>RevLocal = 1 and StopLocal = 1 and Feature2.Bit4 = 0<br>FwdLocal = 1 and RevLocal = 1<br>FwdLocal = 1 and Rev = 1 and Feature.Bit7 = 0<br>RevLocal = 1 and Fwd = 1 and Feature.Bit7 = 0<br>FwdAut = 1 and StopAut = 1 and Feature.Bit4 = 0<br>RevAut = 1 and StopAut = 1 and Feature.Bit4 = 0<br>FwdAut = 1 and RevAut = 1<br>FwdAut = 1 and Rev = 1 and Feature.Bit7 = 0<br>RevAut = 1 and Fwd = 1 and Feature.Bit7 = 0<br>AutModLi = 1 and ManModLi = 1<br>FwdForce = 1 and StopForce = 1<br>RevForce = 1 and StopForce = 1<br>FwdForce = 1 and RevForce = 1<br>SP_LiOp = 1 and SP_IntLi = 1 and SP_ExtLi = 1<br>DvFdOnMan and DvFdOffMan and Feature.Bit15<br>DvFdOnAut and DvFdOffAut and Feature.Bit15 |
| 52 | LocalAct = 1 and LocalSetting = 2 or 4 and SimOn = 1 |

**Note**

The relationship of the pushbutton/switch mode Feature.Bit4 is not taken into account

### Mode switchover error

This error can be output by the block, see the section Error handling (Page 128).

### Invalid or not plausible input commands

This error can be output by the block, see the section Error handling (Page 128).

For the following invalid or not plausible input commands, the control output can be kept or switched to the neutral position. This depends on the function Control priority in the event of invalid or not plausible input commands (Page 186).

| **Operating mode** | **Invalid or not plausible input commands** | **Control reaction with** `Feature2.Bit3` = 1 |
|---|---|---|

| | | |
|---|---|---|
| Local: `Localsetting` = 1 or `Localsetting` = 3 | Pushbutton operation for local mode (`Feature2.Bit4` = 0):<br>`FwdLocal` = 1 and `RevLocal` = 1 or<br>`FwdLocal` = 1 and `StopLocal` = 1 or<br>`StopLocal` = 1 and `RevLocal` = 1<br>Switching mode (`Feature2.Bit4` = 1):<br>`FwdLocal` = 1 and `RevLocal` = 1<br>Enabling direct changeover between forward and reverse (`Feature.Bit7` = 1):<br>`FwdLocal` = 1 and `Rev` = 1 or `RevLocal` = 1 and `Fwd` = 1 | Motor is stopped |
| Local: `Localsetting` = 1 or `Localsetting` = 3 and forcing | `FwdForce` = 1 and `RevForce` = 1 or<br>`FwdForce` = 1 and `StopForce` = 1 or<br>`StopForce` = 1 and `RevForce` = 1 | |
| Forcing and no "local mode" | `FwdForce` = 1 and `RevForce` = 1 or<br>`FwdForce` = 1 and `StopForce` = 1 or<br>`StopForce` = 1 and `RevForce` = 1 | |
| "Automatic mode" and no forcing | Pushbutton operation (`Feature.Bit4` = 0):<br>`FwdAut` = 1 and `RevAut` = 1 or<br>`FwdAut` = 1 and `StopAut` = 1 or<br>`StopAut` = 1 and `RevAut` = 1<br>Switching mode (`Feature.Bit4` = 1):<br>`FwdAut` = 1 and `RevAut` = 1<br>Enabling direct changeover between forward and reverse (`Feature.Bit7` = 1):<br>`FwdAut` = 1 and `Rev` = 1 or `RevAut` = 1 and `Fwd` = 1 | |
| "Manual mode" and no forcing | `FwdMan` = 1 and `RevMan` = 1 or<br>`FwdMan` = 1 and `StopMan` = 1 or<br>`StopMan` = 1 and `RevMan` = 1 | |

### Control system fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 128) section for more on this.

### See also

MotSpdCL block diagram (Page 1535)

MotSpdCL I/Os (Page 1518)

MotSpdCL messaging (Page 1516)

MotSpdCL functions (Page 1496)

MotSpdCL modes (Page 1494)

Description of MotSpdCL (Page 1488)

## 8.5.5 MotSpdCL messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Process messages
- Instance-specific messages

### Process control fault

The following control system error messages can be output:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId1` | SIG 1 | PLC process control message - Failure | $$BlockComment$$<br>Motor feedback error |
| | SIG 2 | PLC process control message - Failure | $$BlockComment$$<br>Motor protection triggered |
| | SIG 3 | PLC process control message - Failure | $$BlockComment$$<br>External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter `CSF`. If it changes to `CSF` = 1,, a process control fault is triggered (`MsgEvId1`, SIG 3).

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 4 | Warning - high | $$BlockComment$$<br>Rbk - high warning limit violated |
| | SIG 5 | Warning - low | $$BlockComment$$<br>Rbk - low warning limit violated |
| | SIG 7 | Alarm - high | Only with Feature Bit5 =1:<br>$$BlockComment$$<br>ER - high warning limit violated |
| | SIG 8 | Alarm - low | Only with Feature Bit5 =1:<br>$$BlockComment$$<br>ER - low warning limit violated |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

With Feature Bit5 =1 SIG 7 and SIG 8 are used for monitoring the difference ER = SP – Rbk. Message class and Event need to be adapted.

### Instance-specific messages

You can use up to three instance-specific messages with this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 6 | PLC process control message - Failure | $$BlockComment$$<br>External message 1 |
| | SIG 7 | PLC process control message - Failure | Only with Feature Bit5 =0 (default): $$BlockComment$$<br>External message 2 |
| | SIG 8 | PLC process control message - Failure | Only with Feature Bit5 =0 (default): $$BlockComment$$<br>External message 3 |

Explanation:

$$BlockComment$$: Content of the instance-specific comment.

With Feature Bit5 =0 SIG 7 and SIG 8 are used for the inputs ExtMsg2.Value and ExtMsg3.Value. Message class and Event need to be adapted.

### Associated values for message instance `MsgEvId1`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | `ExtVa104` |
| 5 | `ExtVa105` |
| 6 | `ExtVa106` |
| 7 | `ExtVa107` |
| 8 | `ExtVa108` |
| 9 | Reserved |
| 10 | Reserved |

The associated values 4 ... 8 are allocated to the parameters `ExtVa104` ... `ExtVa108` , and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

MotSpdCL block diagram (Page 1535)

MotSpdCL I/Os (Page 1518)

MotSpdCL error handling (Page 1512)

MotSpdCL functions (Page 1496)

MotSpdCL modes (Page 1494)

Description of MotSpdCL (Page 1488)

## 8.5.6 MotSpdCL I/Os

I/Os of MotSpdCL

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `AddOn` | AddOn block connector for device specific view | ANY | - |
| `AutModLi*` | 1 = "Automatic mode" via interconnection or SFC (controlled by `ModLiOp` = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `AutModOp*` | 1 = "Automatic mode" via operator (controlled by `ModLiOp` = 0) | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| AV | Input additional analog value, to be connected to AV_Tech of the AV block | ANY | |
| AV_AH_Lim | Limit high alarm | REAL | 95.0 |
| AV_AL_Lim | Limit low alarm | REAL | 5.0 |
| AV_Hyst | Hysteresis for alarm, warning and tolerance limits | REAL | 1.0 |
| AV_TH_Lim | Limit high tolerance | REAL | 85.0 |
| AV_TL_Lim | Limit low tolerance | REAL | 15.0 |
| AV_WH_Lim | Limit high warning | REAL | 90.0 |
| AV_WL_Lim | Limit low warning | REAL | 10.0 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkTypeData* | This input can not be edit or linked and is for internal usage only! | | |
| BypProt | 1 = Bypassing interlock in "local mode" and in "simulation" | BOOL | 0 |
| CSF | 1 = External error (control system fault) Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DvFdOnAut* | 1 = Activation of device infeed in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DvFdOffAut* | 1 = Deactivation of device infeed in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DvFdOnMan* | 1 = Activation of device infeed in "manual mode" | BOOL | 0 |
| DvFdOffMan* | 1 = Deactivation of device infeed in "manual mode" | BOOL | 0 |
| EN | 1 = Called block will be processed | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ER_A_DC* | Reserved | REAL | 0.0 |
| ER_A_DG* | Reserved | REAL | 0.0 |
| ER_AH_En | 1 = Activate alarm (high) for setpoint difference monitoring | BOOL | 1 |
| ER_AH_DFac* | Delay factor at the positive setpoint step changes for incoming alarms at the error signal monitoring ER_AH_Lim | REAL | 0.0 |
| ER_AH_Lim | Alarm limit (high) for setpoint difference monitoring | REAL | 100.0 |
| ER_AH_MsgEn | 1 = Activate messages for alarm (high) for setpoint difference monitoring Messages are only output if you have also enabled the Feature.Bit5 Alarm setpoint difference (Page 182). | BOOL | 1 |
| ER_AL_DFac* | Delay factor at the negative setpoint step changes for incoming alarms at the error signal monitoring ER_AL_Lim | REAL | 0.0 |
| ER_AL_En | 1 = Activate alarm (low) for setpoint difference monitoring | BOOL | 1 |
| ER_AL_Lim | Alarm limit (low) for setpoint difference monitoring | REAL | -100.0 |
| ER_AL_MsgEn | 1 = Activate messages for alarm (low) for setpoint difference monitoring Messages are only output if you have also enabled the Feature.Bit5 Alarm setpoint difference (Page 182). | BOOL | 1 |
| ER_Hyst | Alarm hysteresis for error signal | REAL | 1.0 |
| EventTsIn | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| EventTs2In | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTs2In input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg3 | Binary input for freely selectable message 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa104 | Associated value 4 for messages (MsgEvID1) | ANY | |
| ExtVa105 | Associated value 5 for messages (MsgEvID1) | ANY | |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| ExtVa108 | Associated value 8 for messages (MsgEvID1) | ANY | |
| FactGR | Gear reducing factor | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |
| FaultExt | 1 = External error<br>Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkFwd | 1= Motor feedback running forward | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| FbkRev | 1= Motor feedback running reverse | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| FbkStop | 1= Motor feedback stopped | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| `Feature` | I/O for additional functions (Page 1496)<br>1 = Separate monitoring time for stopping the motor<br>1 = Setpoint with own scale and unit of the parameter | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 13: BOOL<br>• Bit 16: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0<br>• 0<br>• 0<br>• 0 |
| `Feature2` | I/O for additional functions (Page 1496) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| `FwdAut*` | 1 = Activation of forward motor operation in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `FwdChnST` | Signal status of output channel for `Fwd`<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| `FwdForce` | 1 = Forcing activation of motor forward mode | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `FwdLocal` | 1 = Activation of forward motor operation in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `FwdMan*` | 1 = Activation of forward motor operation in "manual mode" | BOOL | 0 |
| `IdleTime*` | Wait time for change of direction or restart in [s] | REAL | 5.0 |
| `Intlock` | 0 = Interlock / Interlock Open is activated.<br>Once the interlock condition has cleared, the block can be operated without reset.<br>1 = Interlock / Interlock Forward is deactivated. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| `Intl_En` | 1 = Parameter `Intlock` is enabled | BOOL | 1 |
| `IntlRev` | 0 = Interlock Reverse is activated.<br>Once the interlock condition has cleared, the block can be operated without reset.<br>1 = Interlock Reverse is deactivated. | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| IntlRevEn | 1 = Parameter IntlRev is enabled | BOOL | 1 |
| LocalLi | 1 = Activate "local mode" via plant signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalOp* | 1 = "Local mode" via operator | BOOL | 0 |
| LocalSetting | Characteristics of Local mode (Page 86) | INT | 0 |
| ManModLi* | 1 = "Manual mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManModOp* | 1 = "Manual mode" via OS operator (controlled by ModLiOp = 0) | BOOL | 1 |
| ModLiOp | Switchover of operating mode between:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Monitor | 1 = Feedback monitoring | BOOL | 1 |
| MonSafePos | 1 = Go to neutral position in the event of monitoring errors  Motors: Stop is the neutral position | BOOL | 1 |
| MonTiDynamic* | Monitoring time for feedback errors or feedback start error after successful operation in [s] | REAL | 3.0 |
| MonTiDyStop* | Monitoring time for feedback stop errors after successful operation in [s] | REAL | 3.0 |
| MonTiStatic* | Monitoring time for feedback errors without operation in [s] | REAL | 3.0 |
| MS_RelOp* | 1= Release for maintenance via OS operator | BOOL | 0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NoFbkFwd | 1= Motor Feedback running forward not present | BOOL | 0 |
| NoFbkRev | 1= Motor Feedback running reverse not present | BOOL | 0 |
| NoFbkStop | 1= Motor feedback stopped not present | BOOL | 1 |
| Occupied | 1 = In use by a batch | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 1496)<br>1 = Operator can set or change the monitoring time for "Control: Start" | STRUCT<br>• Bit 0: BOOL<br>• Bit8: BOOL<br>• Bit 10: BOOL<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1<br>• 1 |
| OS1Perm | I/O for operator permissions (Page 1496)<br>1 = Operator can set or change the monitoring time for "Control: Stop" | STRUCT<br>• Bit 0: BOOL<br>• Bit 9: BOOL<br>• Bit 30: BOOL<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1<br>• 1 |
| Permit | 1 = Activation enable / Activation Forward enable<br>0 = No activation enable / No activation Forward enable | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| PermitSafePos | 1=Motor stop is allowed | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| PermSf_En | 1=Permit stop enabled | BOOL | 1 |
| Perm_En | 1 = Parameter Permit enable | BOOL | 1 |
| PermRev | 1 = Activation Reverse enable0 = No activation Reverse enable | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| PermRevEn | 1 = Parameter PermRev is enabled | BOOL | 1 |
| Protect | 0 = Protective interlock / protective interlock Forward is activated.<br>Once the interlock condition has cleared, the block must be reset<br>1 = Protective interlock / protective interlock Forward is deactivated. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| Prot_En | 1 = Parameter Protect enable | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ProtRev | 0 = Protective interlock Reverse is activated.<br>Once the interlock condition has cleared, the block must be reset<br>1 = Protective interlock Reverse is deactivated. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| ProtRevEn | 1 = Parameter ProtRev enable | BOOL | 1 |
| PulseWidth* | Pulse width of control signal [s] | REAL | 3.0 |
| RapidStp* | Rapid stop for the motor:<br>0 = Motor On<br>1 = Motor Off | BOOL | 0 |
| Rbk | Position feedback for display on OS | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RbkW_DC* | Delay time for incoming warnings [s] | REAL | 0.0 |
| RbkW_DG* | Delay time for outgoing warnings [s] | REAL | 0.0 |
| RbkHyst | Hysteresis for warning limits | REAL | 1.0 |
| RbkOpScale | Limit for scale in bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| RbkUnit | Unit of measure for additional analog value | INT | 0 |
| RbkUnitOut | RbkUnit as output | INT | 0 |
| RbkOpScaleOut | RbkOpScale as output | STRUCT<br>• High: REAL<br>• Low: REAL | <br>• 100.0<br>• 0.0 |
| RbkWH_En | 1 = Enable high warning | BOOL | 1 |
| RbkWH_Lim | Limit high warning | REAL | 90.0 |
| RbkWH_MsgEn | 1 = Enable high warning message | BOOL | 1 |
| RbkWL_En | 1 = Enable low warning | BOOL | 1 |
| RbkWL_Lim | Limit low warning | REAL | 10.0 |
| RbkWL_MsgEn | 1 = Enable low warning message | BOOL | 1 |
| RevAut* | 1 = Activation of reverse motor operation in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RevChnST | Signal status of output channel for Rev<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| RevForce | 1 = Force activation of reverse motor operation | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RevLocal | 1 = Activation of reverse motor operation in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RevMan* | 1 = Activation of reverse motor operation in "manual mode" | BOOL | 0 |
| RstLi* | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | 1 = Reset via operator | BOOL | 0 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | |
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | |
| SimAV* | Additional value used for SimOn = 1 | REAL | 0.0 |
| SimAV_Li | Additional analog value that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimRbk* | Position feedback used for SimOn = 1 | REAL | 0.0 |
| SimRbkLi | Position feedback used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_DnRaLim* | Limit (low) for the gradient of the setpoint [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 100.0 |
| SP_Ext | external setpoint - (to interconnection) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP_ExtLi | 1 = Select external setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtOp* | 1 = Select external setpoint (via operator) | BOOL | 0 |
| SP_HiLim | Limit (high) of setpoint | REAL | 100.0 |
| SP_LoLim* | Limit (low) of setpoint;<br>if Feature2.Bit7 = 1<br>then SP_LoLim is limited and written back to SP_Off. | REAL | 0.0 |
| SP_Int* | Internal setpoint for operation | REAL | 1.0 |
| SP_IntLi | 1 = Select internal setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_IntOp* | 1 = Select internal setpoint (via operator) | BOOL | 1 |
| SP_LiDnRaLim | Linkable limit (low) for the gradient of the setpoint [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_LiOp | Select setpoint source (internal/external):<br>1 = Via interconnection<br>0 = Via operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_LiRmpTime | Linkable time [Time unit] for gradient limitation in ramp mode duration<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_LiUpRaLim | Linkable gradient limit (high) for the setpoint [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_Off | Setpoint if the motor is off | REAL | 0.0 |
| SP_OpScale | SP bar limit display for OS | ScaVal | |
| SP_OutChnST | Signal status of output channel SP_Out<br>Should be connected to an output channel block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| SP_RateOn* | 1 = Activate limitation of setpoint gradients | BOOL | 0 |
| SP_RmpModTime | 1 = Use time values (SP_RmpTime, SP_LiRmpTime)<br>0 = Use gradient limits (SP_DnRaLim, SP_LiDnRaLim, SP_UpRaLim, SP_LiUpRaLim) | BOOL | 0 |
| SP_RmpOn* | 1 = Activate setpoint ramp to target setpoint SP_RmpTarget | BOOL | 0 |
| SP_RmpTarget* | Target setpoint for setpoint ramp | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP_RmpTime* | Time [Time unit] for setpoint ramp from current SP up to SP_RmpTarget or time [Time unit] for gradient limitation in ramp mode duration<br>Time unit depends on TimeFactor | REAL | 0.0 |
| SP_TrkExt | 1 = Bumpless switchover from external to internal setpoint active | BOOL | 0 |
| SP_Unit | Operator inputs | INT | 0 |
| SP_UpRaLim* | Gradient limit (high) for the setpoint [Unit/(Time unit)] Time unit depends on TimeFactor | REAL | 100.0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| StopAut* | 1 = Stopping the motor in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopForce | 1 = Force motor stop | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopLocal | 1 = Stopping the motor in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopMan* | 1 = Stopping the motor in "manual mode" | BOOL | 0 |
| TimeFactor | Time unit:<br>0 = Seconds<br>1 = Minutes<br>2 = Hours | INT | 0 |
| Trip | 1 = Motor is in "good" state | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| UserAna1 | Analog auxiliary value 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UA1unit | Unit of measure for analog auxiliary value 1 | INT | 0 |
| UserAna2 | Analog auxiliary value 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UA2unit | Unit of measure for analog auxiliary value 2 | INT | 0 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| WarnTiAut* | Prewarning of motor start in "automatic mode" in [s] | REAL | 0.0 |
| StateTrkOn | Initiate tracking of state | BOOL | 0 |
| WarnTiMan* | Prewarning of motor start in "manual mode" in [s] | REAL | 0.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutAct | 1 = "Automatic mode" enabled<br>0 = "Manual mode" is enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutoST | 1 = Worst signal status of automatic commands | BYTE | 16#80 |
| AV_OpScale | Limit for scale in AV bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| AV_Out | Output additional analog value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| AV_Unit | Unit of measure for additional analog value | INT | 0 |
| DvFdAct | 1 = Device infeed enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| CurrMon | Remaining monitoring time [s] | DINT | 0 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ER | Error signal | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ER_A_DCOut | Effective delay time [s] for incoming alarms at the error signal monitoring | REAL | 0.0 |
| ER_AH_Act | 1 = Alarm limit (high) for control deviation violated. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ER_AL_Act | 1 = Alarm limit (low) for control deviation violated. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ER_AH_LimOut | ER_AH_Lim as output | REAL | 0.0 |
| ER_AL_LimOut | ER_AL_Lim as output | REAL | 0.0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see MotSpdCL error handling (Page 1512) | INT | -1 |
| FaultExtOut | FaultExt as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkFwdOut | 1= Motor feedback running forward | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkRevOut | 1= Motor feedback running reverse | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkStopOut | 1= Motor feedback stopped | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Fwd | 1 = Control of motor forward | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| GrpErr | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IntlockOut | Intlock as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IntlRevOut | IntlRev as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalAct | 1 = "Local mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LockAct | 1 = Interlock (Intlock, Permit, Protect) or Trip is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManAct | 1 = "Manual mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MonDynErr | 1 = Feedback error or feedback start error due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MonDynStopErr | 1 = Feedback stop error due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MonStaErr | 1 = Feedback error due to unexpected feedback change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message acknowledgement status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | Alarm error 1 (output ERROR of first ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| OosAct | 1 = Block is "out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1Perm | DWORD | 16#FFFFFFFF |
| OS1PermLog | Display of OS1Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| P_Fwd | 1 = Pulse signal for starting the motor in forward | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Rev | 1 = Pulse signal for starting the motor in reverse | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Rst | 1= Pulse output for reset<br>The parameter persists for one cycle after a reset. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Stop | 0 = Pulse signal for stopping the motor | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PermitOut | Permit as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PermitSafePosOut | PermitSafePos as output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| PermRevOut | PermRev as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ProtectOut | Protect as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ProtRevOut | ProtRev as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| R_StpAct | 1 = Rapid stop of the motor is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RbkOut | Output of readback value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RbkWH_Act | 1 = Warning (high) enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RbkWL_Act | 1 = Warning (low) enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToReset | 1 = Ready for reset via RstLi input or commands in "local", "automatic", or "manual mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToStart | 1 = Active start readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Rev | 1 = Control of motor: Reverse | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RunFwd | 1 = Motor is running forwards | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RunRev | 1 = Motor is running in reverse | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_DnRaAct | 1 = Negative gradient limiting of setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtAct | 1 = External setpoint is active<br>0 = Internal setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_HiAct | 1 = Limit (high) for setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_LoAct | 1 = Limit (low) for setpoint has been reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_Out | Setpoint used by controller | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_Out2 | Additional setpoint, without gear reducing factor | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_RateTarget | Target setpoint for the gradient limitation | REAL | 0.0 |
| SP_UpRaAct | 1 = Positive gradient limiting of setpoint is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_RemRT | Remaining ramp time of the setpoint | REAL | 0.0 |
| Starting | 1 = Motor will start | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Status1 | Status word 1 (Page 1488) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1488) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1488) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 1488) | DWORD | 16#00000000 |
| Status5 | Status word 5 (Page 1488) | DWORD | 16#00000000 |
| Status6 | Status word 6 (Page 1488) | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Stop | 1 = Motor is stopping | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopCtrl | Control output to Stop the motor (go to Safe position) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Stopping | 1 = Motor will stop | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| TripOut | Trip as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| UA1unitOut | UA1 as output | INT | 0 |
| UA2unitOut | UA2 as output | INT | 0 |
| UserAna1Out | UserAna1 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| UserAna2Out | UserAna2 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| ResyncST | Worst Signal Status of inputs for resynchronization | BYTE | 16#80 |
| WarnAct | 1 = Prewarning for motor start active (parameters WarnTiAut and WarnTiMan) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

## See also

MotSpdCL block diagram (Page 1535)
MotSpdCL messaging (Page 1516)
MotSpdCL modes (Page 1494)
Setpoint specification with separate display area and custom unit (Page 182)
Error handling (Page 128)

## 8.5.7 MotSpdCL block diagram

### MotSpdCL block diagram

A block diagram is not provided for this block.

### See also

MotSpdCL I/Os (Page 1518)

MotSpdCL messaging (Page 1516)

MotSpdCL error handling (Page 1512)

MotSpdCL functions (Page 1496)

MotSpdCL modes (Page 1494)

Description of MotSpdCL (Page 1488)

## 8.5.8 Operator control and monitoring

### 8.5.8.1 MotSpdCL views

### Views of the MotSpdCL block

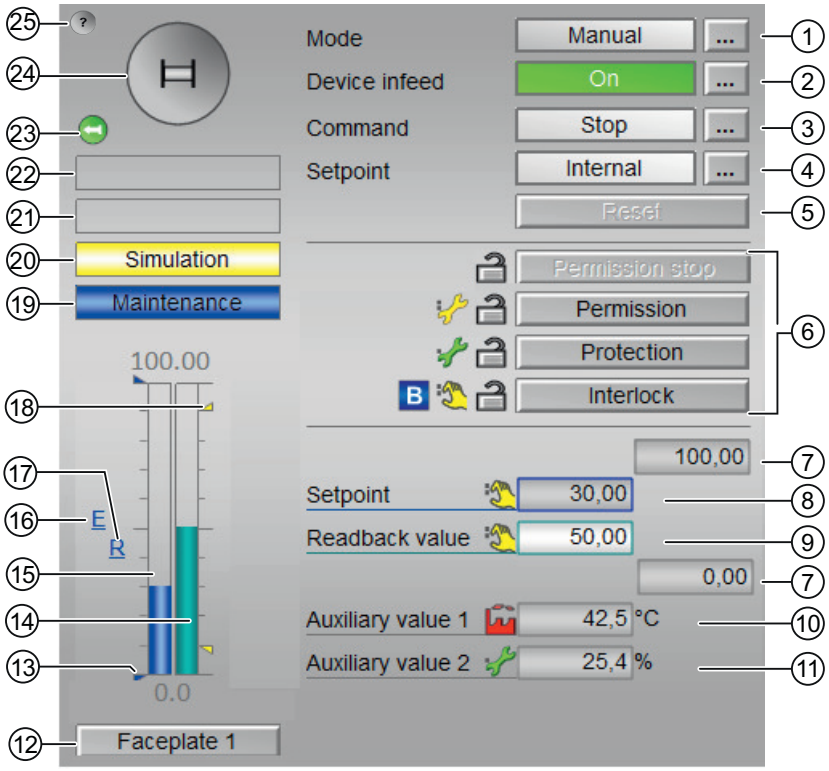
The block MotSpdCL provides the following views:

- MotSpdCL standard view (Page 1536)
- Alarm view (Page 336)
- Limit view of motors (Page 328)
- MotSpdCL limit view for readback values (Page 1546)
- Trend view (Page 341)
- Ramp view (Page 334)
- MotSpdCL parameter view (Page 1548)
- MotSpdCL preview (Page 1542)
- MotSpdCL AddOn view (Page 1552)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for MotSpdCL (Page 1552)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 8.5.8.2 MotSpdCL standard view

#### MotSpdCL standard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual and automatic mode for motors, valves and dosers (Page 82)
- Manual and automatic mode for motors, valves and dosers (Page 82)
- Local mode (Page 86)
- Out of service (Page 77)

See also the section Switching operating states and operating modes (Page 289) for information on switching the operating mode.

#### (2) Separate switching on/off of the device infeed or the inverter enable

This area shows you the default operating state for the device infeed or the inverter enable. The following states can be shown and executed here:

- "On"
- "Off"

If the inverter enable is parameterized (`Feature.Bit15` = 1 and `Feature.Bit8` = 1), the text "Device infeed" is replaced by "Inverter enable".

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

If the text is configured for these commands, it is displayed as status text and as button labels for command selection. You can find additional information on this in the section Labeling of buttons and text (Page 235).

### (3) Starting and stopping the motor

This area shows you the default operating state for the variable motor. The following states can be shown and executed here:

- "Start |→"
- "Start ←|"
- "Stop"
- "Rapid stop"

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

If the text is configured for these commands, it is displayed as status text and as button labels for command selection. You can find more information about this in section Labeling of buttons and text (Page 235).

### (4) Switching the setpoint internal/external

This area shows how to specify the setpoint. The setpoint can be specified as follows:

- By the application ("External", CFC/SFC)
- By the user directly in the faceplate ("Internal").

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the setpoint specification.

You can find additional information on this in the section Setpoint specification - internal/ external (Page 136).

### (5) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in the section Resetting the block in case of interlocks or errors (Page 47).

### (6) Operating range for the interlock functions of the block

This display is only visible when the corresponding block input is connected.

You can use this button to control the interlock functions of the block. You can find additional information on this in the section Interlocking functions (Page 107).

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 113)), for example:

- Signal status (see Forming and outputting the signal status for technologic blocks (Page 117)), for example:

- Bypass information (see Forming the group status for interlock information (Page 113)):

If the text is configured for the interlocks, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (7) High and low scale range for the setpoint

This area is already set and cannot be changed.

### (8) Display and change the setpoint including signal status

This area shows the current setpoint with the corresponding signal status.

Refer to the Changing values (Page 291) section for information on changing the setpoint. The setpoint specification also needs to be set to "Internal" for this block.

### (9) Displaying the readback value

This area shows you the current readback value with the corresponding signal status.

### (10) and (11) Display of auxiliary values

This display is only visible when the corresponding block input is connected.

You can use this area to display two auxiliary values that have been configured in the engineering system (ES). You can find additional information on this in the section Displaying auxiliary values (Page 239).

### (12) Navigation button for switching to the standard view of any faceplate

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234).

### (13) Displaying the limits

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the engineering system (ES).

### (14) Bar graph for the readback value

This area shows you the current readback value in the form of a bar graph. You can also see the high and low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (15) Bar graph for the setpoint

This area shows the current setpoint in the form of a bar graph. You can also see the high and low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (16) Display of external setpoint

This display [E] is only visible when you have selected "Internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

### (17) Display for the target setpoint of the setpoint ramp

This display [R] shows the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 334).

### (18) Limit display

These colored triangles show you the configured limits in the respective bar graph.

For more information, refer: Bar graph (Page 376)

### (19) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this topic is available in sections Release for maintenance (Page 70) Display area for block states.

### (20) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"
- "Delay"

You will find more detailed information on this in the sections Simulating signals (Page 64) and Display of delay times (Page 288).

### (21) Display area for block states

This area provides additional information on the operating state of the block:

- "Motor protection"
- "External error"
- "Status error"
- "Control error"
- "Internal error"
- "Changeover error"
- "Invalid Signal"

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 104) , Error handling (Page 128) (section "Invalid input signals" and "Mode switchover error") and Motor protection function (Page 106).

### (22) Display area for block states

This area provides additional information on the operating state of the block:

- "Forced stop"
- "Forced start |→"
- "Forced start ←|"
- "Request 0/1": Reset is expected but reset is not possible from faceplate. For more details, refer:Reset of errors during Request 0/1 (Page 382)
- "SP ramp active"

You can find additional information on this in the section Forcing operating modes (Page 45).

### (23) Automatic preview

This display is only visible in "manual mode", in "local mode", or with a reset request in "automatic mode", when the current output signals are not identical to the control in "automatic mode".

The display shows what state the motor would assume if you switched from "manual" or "local" mode to "automatic mode", or performed a reset to "automatic mode".

### (24) Status display of the motor

The current status of the motor is graphically displayed here.

You can find more information about this in section Block icon for MotSpdCL (Page 1552)

### (25) Mini preview for Resynchronization

Displays the mini preview for Resynchronizing the signal state.

The mini preview displays the following states:

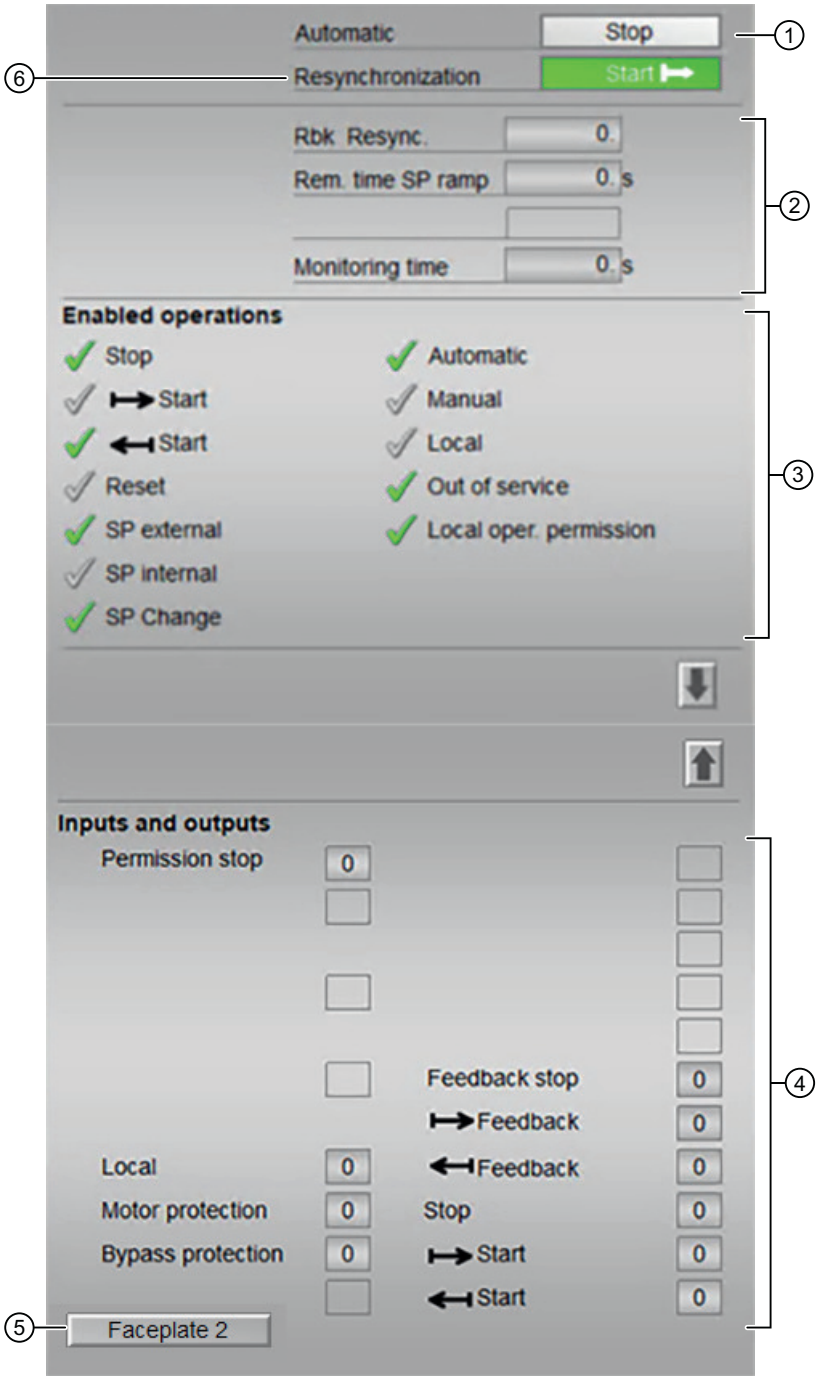
- Fwd
- Rev
- Stop
- No Cmd

**Note**

Mini preview for resynch is visible only if resynchronization command is different to control output value.

### 8.5.8.3 MotSpdCL preview

**Preview of MotSpdCL**



### (1) Automatic preview

This area shows you the block status after it has switched from the "Manual" mode or "Local" mode to the "Automatic" mode.

If the block is in "Automatic" mode, the current block state is displayed.

**Note**

- If the valve is closed and interlock locking is active (`SafePos`=0 Close), the automatic preview does not show the current status "Closed", instead it shows the active automatic command "Open".
- If the valve is in the end position and the automatic stop command is active, the automatic preview does not show the current status "Closed", but the active automatic command "Stop".

This area also displays the worst signal status of the following automatic commands:

- `FwdAut`
- `RevAut`
- `StopAut`
- `SP_Ext`

### (2) Preview area

This area shows you a preview of the following values:

- "Rem. time SP ramp" : Remaining time to reach the ramp target value.
- "Channel Setpoint": Setpoint from the output channel block.
- "Monitoring time": Display of the remaining monitoring time.
- "Rbk Resync": Shows the input Rbk value for Resynchronization

### (3) Enable operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "Stop": You can stop the motor.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Start |→ ": You can start the motor.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Start ←|": You can start the motor.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Reset": You can reset the motor after interlocks or errors.
- "SP external": You can feedforward the external setpoint.
- "SP internal": You can feedforward the internal setpoint.
- "Change SP": You can change the setpoint.
- "Automatic": You can switch to "automatic mode".
- "Manual": You can switch to "manual mode".
- "Local": You can switch to "local mode".
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operator permission": Use the ← button to switch to the standard view of the OpStations block. You can find additional information on this in the section Operator control permissions (Page 285).
- "Device infeed On": You can activate the device infeed
- "Device infeed Off": You can deactivate the device infeed

### (4) Displaying current control signals

This area shows the most important parameters for this block with the current selection:

- "Permission" (`Feature2.Bit16` =0):
  This display is only visible when the corresponding block input is connected.
  - 0 = No OS release for energizing motor
  - 1 = Enable for "opening"/"closing" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- " Permission Fwd" (`Feature2.Bit16` =1):
  This display is only visible when the corresponding block input is connected.
  - 0 = No OS release for energizing motor
  - 1 = Enable for "Start forward" from the neutral position

- " Permission Rev" (`Feature2.Bit16` =1):
  This display is only visible when the corresponding block input is connected.
  - 0 = No OS release for energizing motor
  - 1 = Enable for "Start reverse" from the neutral position
  - to reset the block
  - 1 = "Good" state
- "Protection" (`Feature2.Bit16` =0):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection Fwd" (`Feature2.Bit16` =1):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state
- "Protection Rev" (`Feature2.Bit16` =1):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state
- "Interlock" (`Feature2.Bit16` =0):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock Fwd" (`Feature2.Bit16` =1):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state
- "Interlock Rev" (`Feature2.Bit16` =1):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

- "Local": 1 = Control signal for "Local mode" (LocalLi) is active
- "Motor protection": 1 = Motor is in "good" state
- "Interlock deact.":
  - 0 = Bypass disabled
  - 1 = Bypassing interlock in "local mode" and in "simulation"
- "Local stop": 1= Block is operated in "local mode"
- "Local start |→": 1= Block is operated in a controlled manner in "local mode"
- "Local start ←|": 1= Block is operated in a controlled manner in "local mode"
- "Feedback |→": 1 = Motor has started and is running
- Feedback←|: 1 = Motor has started and is running
- "Feedback stop" : 1 = Motor has stopped
- "Stop": 1 = Stop motor
- "Start | →" 1 = Start motor
- "Start ←|": 1 = Start motor
- "Device infeed": 1 = Enable device infeed
- "Channel Start →":Signal from the output channel block for "Start"
- "Channel Start ←":Signal from the output channel block for "Start"
- "Permission stop": 1 = Motor can not be stopped in manual (by operator) or automatic (by logic) mode

**(5) Navigation button for switching to the standard view of any faceplate**

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

For more information, see section Opening additional faceplates (Page 234).

**(6) Resynchronization**

Displays the Resynchronization state and Resynchronization Signal status based on feedback and readback values.

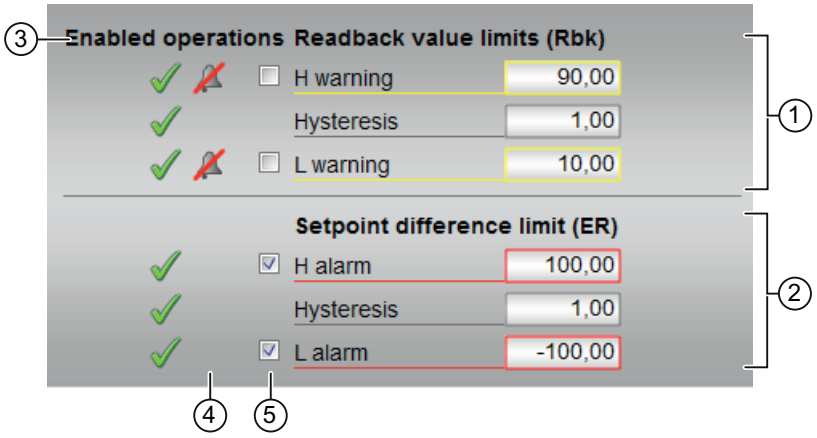
### 8.5.8.4 MotSpdCL limit view for readback values

**Limit view for readback values of MotSpdCL**

Several values are set in this view by default:

- Readback value limits

The toolbars of the faceplate and the block icon indicate when the limits are reached or violated.




### (1) Displaying and changing the limits for the readback value

In this area, you can enter the limits for the readback value. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H warning": Warning high
- "Hysteresis"
- "L warning": Warning low

### (2) Displaying and changing the limits for the setpoint difference

In this area, you can enter the limits for the setpoint difference. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H alarm": Alarm high
- "Hysteresis"
- "L alarm": Alarm low

### (3) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

### (4) "Message suppression/delay"

Message suppression indicates whether or not the suppression of the associated message in the AS block is activated with the `xx_MsgEn` parameter. The output of messages is not suppressed when the block is installed (all `xx_MsgEn` parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

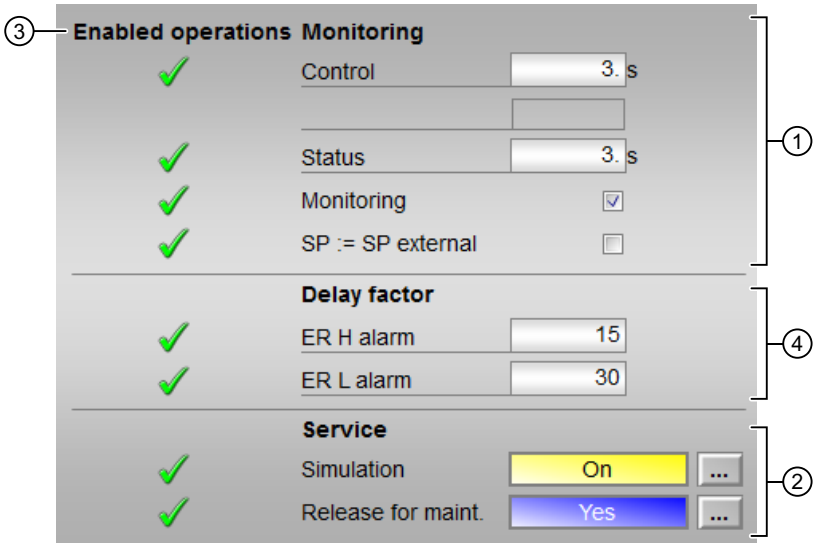
Alarm delays are also displayed in this position; for more on this see section Area of application of the alarm delays (Page 225).

### (5) Suppress messages

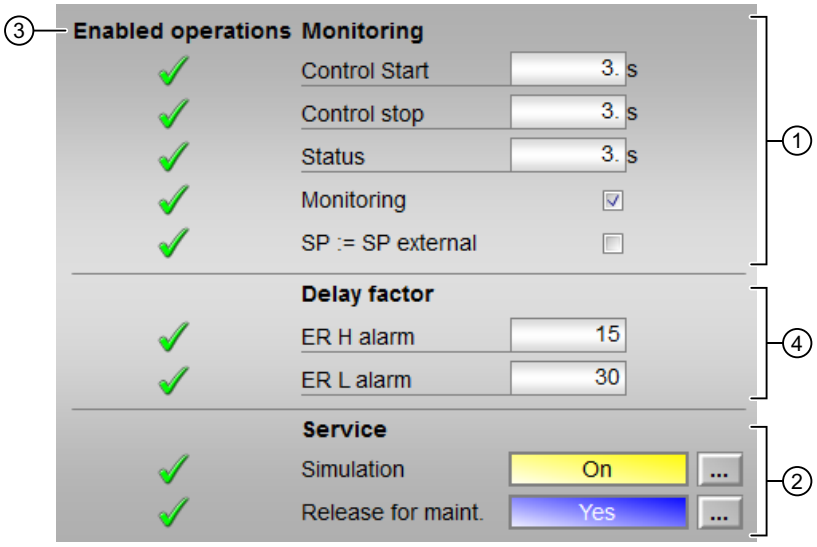
You can enable / disable messages by setting the check mark.

## 8.5.8.5 MotSpdCL parameter view

### Parameter view of MotSpdCL




Parameter view of MotSpdCL with `Feature.Bit13` = 0




Parameter view of MotSpdCL with `Feature.Bit13` = 1

## (1) Monitoring

In this area, you change parameters and therefore influence the motor. Refer to the Changing values (Page 291) section for more on this.

You can influence the following parameters:

- `Feature.Bit13` **= 0**
  - "Control": Monitoring time during startup and shutdown of the motor (dynamic)
- `Feature.Bit13` **= 1**
  - "Control stop": Monitoring time during shutdown of the motor (dynamic)
  - "Control start": Monitoring time during startup of the motor (dynamic)

**"Status" monitoring**

You can monitor time during permanent operation of the motor (static).

**Enable monitoring**

You can enable monitoring by selecting the check box (☑)

You can find additional information on this in the section Monitoring the feedbacks (Page 104).

**Activating bumpless switchover**

"SP := SP external": ☑ Bumpless switchover of setpoint from external to internal. The internal setpoint is tracked to the external one.

### (2) Service

You can select the following functions in this area:

- Simulation
- Release for maintenance

Refer to the Switching operating states and operating modes (Page 289) section for more on this.

You can find information on this area in the section:

- Simulating signals (Page 64)
- Release for maintenance (Page 70)

### (3) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

### (4) Delay factor

In this area, you can change the following parameters:

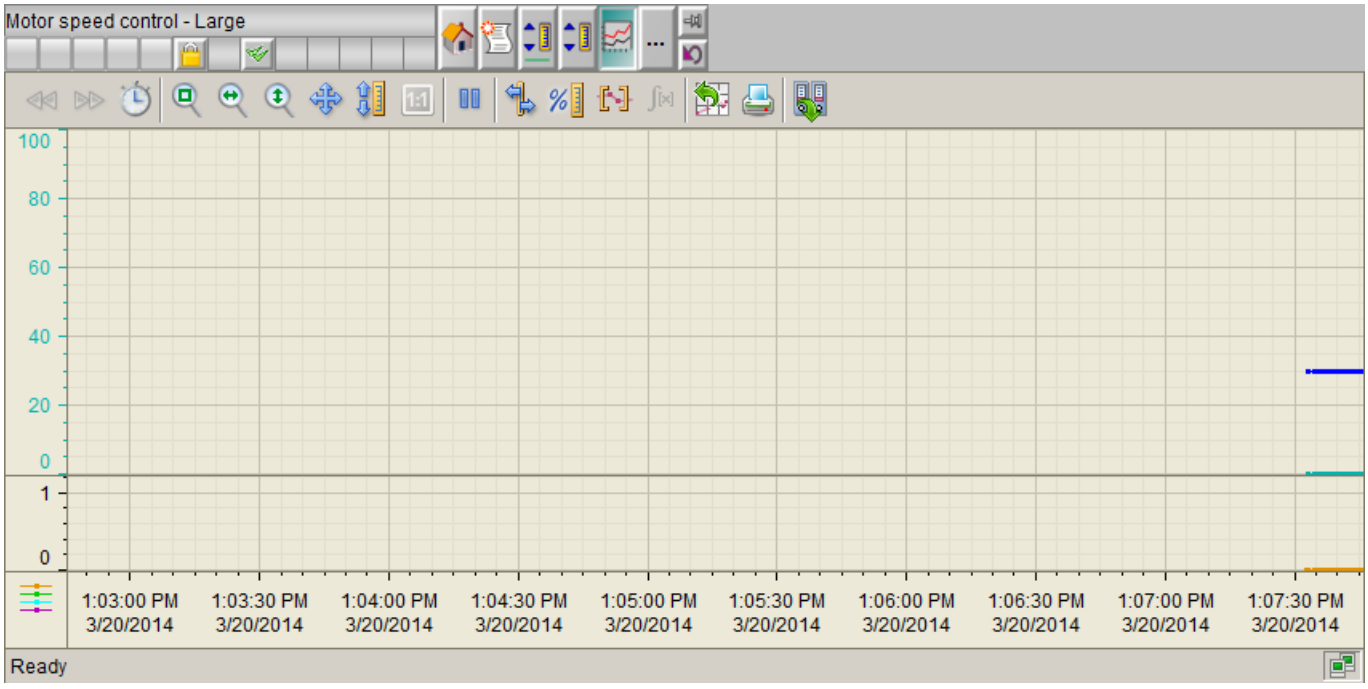
- "ER H alarm": Delay factor at the positive setpoint step changes for incoming alarms at the control deviation monitoring `ER_AH_Lim`.
- "ER L alarm": Delay factor at the negative setpoint step changes for incoming alarms at the control deviation monitoring `ER_AL_Lim`.

## 8.5.8.6 MotSpdCL trend view

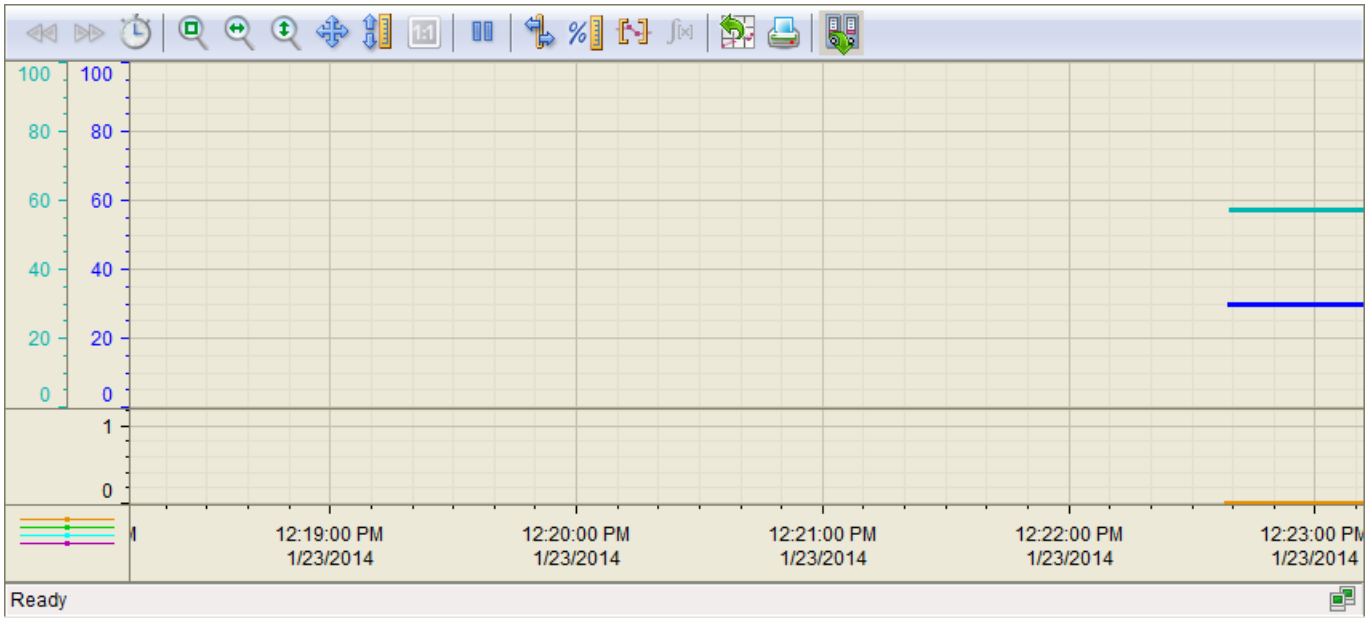
### Trend view of MotSpdCL

You can find general information in section Trend view (Page 341).

Depending on `Feature.Bit16` (process value with separate scale range) either one or two value axes are shown in the trend view.




- Process value with separate scale range



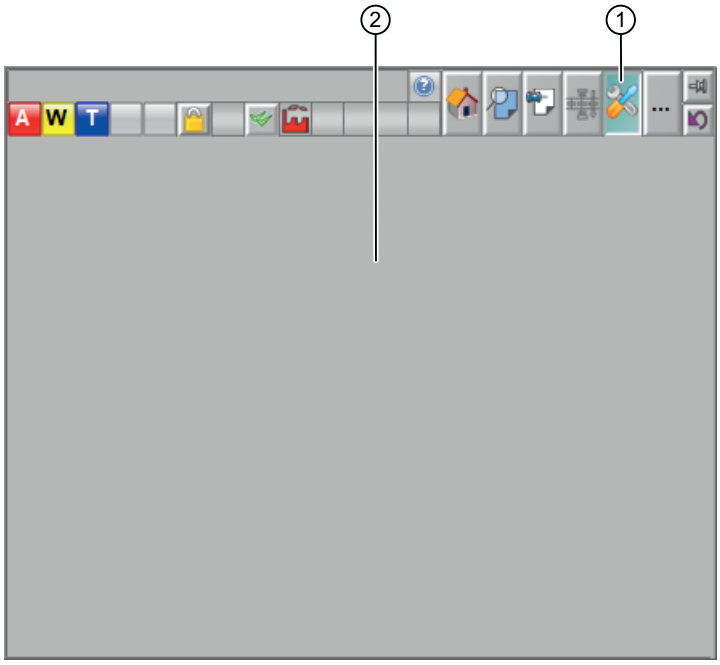

- Process value without separate scale range

**Note**

This function only has an effect when `SP_Out` is configured in "TrendConfiguration1" and `PV_In` is configured in "TrendConfiguration2" in the block icon (default).

### 8.5.8.7 MotSpdCL AddOn view

#### MotSpdCL AddOn view




(1) AddOn view icon

(2) Work space of AddOn view: The view of the faceplate can be customized as per the operator needs.

For more information, on AddOn view, refer to AddOn View for function blocks (Page 343)

### 8.5.8.8 Block icon for MotSpdCL

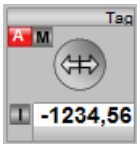
#### Block icons for MotSpdCL

A variety of block icons are available with the following functions:

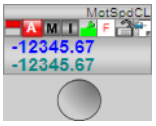
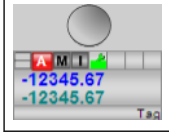
- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as control system faults
- Operating modes
- Internal and external setpoint specification
- Signal status, release for maintenance
- Displays for bypassing interlocks
- Interlocks
- Memo display

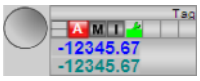
- Motor state display
- Setpoint (blue, with decimal places)
- Feedback value (green, with decimal places)

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | Direction display inverted to symbol 1 |
|  | 4 | Direction display inverted to symbol 2 |
|  | 5 | "M" symbol with small direction display |
|  | 6 | "M" symbol with small direction display |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |

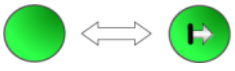
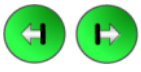
| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 3 | |
|  | 4 | |
|  | 5 | |
|  | 6 | |
|  | 7 | |
|  | 8 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

## Motor state display

The following motor states are shown here:

| Icon | Meaning |
|---|---|
|  | Motor started (motor symbol changes) |
|  | The motor is running |
|  | Motor stopped (motor symbol changes) |
|  | Motor idle |
|  | Error at motor (monitoring error, Motor protection) |

| Icon | Meaning |
|---|---|
|  | Motor out of service |
|  | “No Cmd” State |

# 8.6 MotSpdL - Two-speed motor

## 8.6.1 Description of MotSpdL

### Object name (type + number) and family

Type + number: FB 1856

Family: Drives

### Area of application for MotSpdL

The block is used for the following applications:

- Control of two-speed motors

### How it works

The block is used to control motors with two speeds. Various inputs are available for controlling the motor. You can add monitoring of a maximum of two feedbacks produced by the contactor relay and a switching mode with which the speed change can be performed.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB3x). The block is also installed automatically in the startup OB (OB100).

Further addressing is not required.

For the MotSpdL block, the Advanced Process Library contains a template for process tag types as an example with an application scenario for this block.

Example of process tag types:

- Two-speed motor (Motor2Speed) (Page 2860)

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section MotSpdL I/Os (Page 1578).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |

| Status bit | Parameter |
|---|---|
| 2 | SimOn |
| 3 | OosAct.Value |
| 4 | OosLi.Value |
| 5 | 0 = ManAct.Value<br>1 = AutoAct.Value |
| 6 | LocalAct.Value |
| 7 | 0: Open padlock in the block icon<br>1: Closed padlock in the block icon<br><br>In case of Feature2.Bit16 = 1 then a closed padlock is displayed if Interlock Open OR Interlock Close is active. |
| 8 | Spd1.Value |
| 9 | Motor is stopped |
| 10 | Spd2.Value |
| 11 | MonStaErr.Value |
| 12 | MonDynErr.Value |
| 13 | BypProt |
| 14 | Invalid signal status |
| 15 | Mode switchover error |
| 16 | 1 = Intlock is active |
| 17 | 1 = Permit is active |
| 18 | 1 = Protect is active |
| 19 | Trip.Value |
| 20 | Display „Forced speed1“ |
| 21 | Display „Forced stop“ |
| 22 | Display „Forced speed2“ |
| 23 | "Interlock" button is enabled |
| 24 | Reset request in automatic preview |
| 25 | WarnAct:Value,<br>IdleTime or<br>SwOverTi active |
| 26 | Bypass information for display in block icon |
| 27 | Automatic preview for (Speed1) |
| 28 | Automatic preview for (Stop) |
| 29 | Automatic preview for (Speed2) |
| 30 | "Permission" button is enabled |
| 31 | "Protection" button is enabled |

## Status word allocation for Status2 parameter

| Status bit | Parameter |
|---|---|
| 0 | MsgLock |
| 1 | AV_AH_Act.Value |

| Status bit | Parameter |
|---|---|
| 2 | AV_WH_Act.Value |
| 3 | AV_TH_Act.Value |
| 4 | AV_TL_Act.Value |
| 5 | AV_WL_Act.Value |
| 6 | AV_AL_Act.Value |
| 7 | AV_AH_En OR Feature.Bit23 |
| 8 | AV_WH_En OR Feature.Bit23 |
| 9 | AV_TH_En OR Feature.Bit23 |
| 10 | AV_TL_En OR Feature.Bit23 |
| 11 | AV_WL_En OR Feature.Bit23 |
| 12 | AV_AL_En OR Feature.Bit23 |
| 13 | AV_AH_MsgEn |
| 14 | AV_WH_MsgEn |
| 15 | AV_TH_MsgEn |
| 16 | AV_TL_MsgEn |
| 17 | AV_WL_MsgEn |
| 18 | AV_AL_MsgEn |
| 19 | 1 = No impact of input signals on "local mode" with LocalSetting = 0, 2, 4 |
| 20 | Motor is stopped |
| 21 | Motor stops at speed 1 |
| 22 | Motor stops at speed 2 |
| 23 | Motor starts at speed 1 |
| 24 | Motor running at speed 1 |
| 25 | Motor starts at speed 2 |
| 26 | Motor running at speed 2 |
| 27 | Error when stopping motor |
| 28 | Error with motor speed 1 |
| 29 | Error with motor speed 2 |
| 30 | Display for interlocks in block icon |
| 31 | MS_RelOp |

### Status word allocation for Status3 parameter

| Status bit | Parameter |
|---|---|
| 0 | Delay of the AV_AH_Lim message |
| 1 | Delay of the AV_WH_Lim message |
| 2 | Delay of the AV_TH_Lim message |
| 3 | Delay of the AV_TL_Lim message |
| 4 | Delay of the AV_WL_Lim message |
| 5 | Delay of the AV_AL_Lim message |
| 6 | Collection of message delays |
| 7 | Resynchronization preview for Speed1 |

| Status bit | Parameter |
|---|---|
| 8 | Resynchronization preview for Speed2 |
| 9 | Resynchronization preview for Stop |
| 10 | Resynchronization: Display preview |
| 11 | Resynchronization: Display small preview symbol in standard view |
| 12 - 13 | Not used |
| 14 | Resynchronization with state “No Cmd” |
| 15 | Current monitoring time is visible |
| 16 | `MonDynStopErr.Value` |
| 17 | Not used |
| 18 | `SimLiOp.Value` |
| 19 | 1 = Enable for rapid stop (`Feature` Bit Enabling rapid stop via faceplate (Page 179)) |
| 20 | 1 = Input parameter `Spd1ChnST` is interconnected |
| 21 | 1 = Input parameter `Spd2ChnST` is interconnected |
| 22 | Not used |
| 23 | Command for rapid stop |
| 24 | Command for starting > the motor |
| 25 | Command for starting >> the motor |
| 26 | Show automatic preview in the standard view |
| 27 | Internal error |
| 28 | `GrpErr.Value` |
| 29 | `RdyToStart.Value` |
| 30 | Auxiliary value 1 visible |
| 31 | Auxiliary value 2 visible |

## Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 | Effective signal 1 of the message block connected via `EventTsIn` |
| 1 | Effective signal 2 of the message block connected via `EventTsIn` |
| 2 | Effective signal 3 of the message block connected via `EventTsIn` |
| 3 | Effective signal 4 of the message block connected via `EventTsIn` |
| 4 | Effective signal 5 of the message block connected via `EventTsIn` |
| 5 | Effective signal `6` of the message block connected via `EventTsIn` |
| 6 | Effective signal 7 of the message block connected via `EventTsIn` |
| 7 | Effective signal 8 of the message block connected via `EventTsIn` |
| 8 | `AV` not connected |
| 9 | Motor protection display (`Trip.Status` ≠ 16#FF) |
| 10 | 1 = Input parameter `FbkSpd1` is connected |
| 11 | 1 = Input parameter `FbkSpd2` is connected |
| 12 | 1 = Input parameter FbkStop is connected |
| 13 - 21 | Not used |

| Status bit | Parameter |
|---|---|
| 22 | External error generated by FaultExt or external control system fault CSF with set `Feature.Bit18` Activating error state for external process control error CSF (Page 162) |
| 23 | Hidden bypass signal in Permit |
| 24 | Hidden bypass signal in interlock |
| 25 | Hidden bypass signal in Protect |
| 26 | `Feature2.Bit2`: Separate bypass signal |
| 27 | `Feature2.Bit16`: Separate interlock for each direction, speed or position |
| 28 | Hidden bypass signal in Permit Close |
| 29 | Hidden bypass signal in Interlock Close |
| 30 | Hidden bypass signal in Protect Close |
| 31 | Separate monitoring of shutdown of the motor (`Feature.Bit13`) |

### Status word allocation for `Status5` parameter

| Status bit | Parameter |
|---|---|
| 0 - 7 | Effective signal 9..16 of the message block connected via `EventTsIn` |
| 8 - 15 | Not used |
| 16 | "Permission Open" button is enabled |
| 17 | "Permission Close" button is enabled |
| 18 | "Interlock Open" button is enabled |
| 19 | "Interlock Close" button is enabled |
| 20 | "Protect Open" button is enabled |
| 21 | "Protect Close" button is enabled |
| 22 | 1 = Permission Close is active |
| 23 | 1 = Intlock Close is active |
| 24 | 1 = Protect Close is active |
| 25 | 1 = "Permission stop" button is enabled |
| 26 | 1 = Permission stop is active |
| 27 | Hidden bypass signal in Permission stop |
| 28 | Not used |
| 29 | 1= Suppressed messages |
| 30 - 31 | Not used |

### See also

MotSpdL functions (Page 1563)

MotSpdL messaging (Page 1577)

MotSpdL block diagram (Page 1590)

MotSpdL error handling (Page 1574)

MotSpdL modes (Page 1561)

## 8.6.2 MotSpdL modes

### MotSpdL modes

The block can be operated using the following modes:

- Local mode (Page 86)
- Automatic mode (Page 82)
- Manual mode (Page 82)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions. This includes, for example, the parameters for mode changes.

### "Local mode"

You can find general information on "Local mode", switching modes and Bumpless switchover in the Local mode (Page 86) section.

Motor actions you can control in local mode:

- starting with speed 1 (`Spd1Local` = 1)
- starting with speed 2 (`Spd2Local` = 1)
- stopping (`StopLocal` = 1).

A motor operated in "local" mode is controlled either by "local" signals (input parameters `Spd1Local` = 1, `Spd2Local` = 1 and `StopLocal` = 1) or feedback signals (input parameters `FbkSpd1` = 1, `FbkSpd2` = 1 or `FbkStop` =1). Configuration takes place via the input parameter `LocalSetting`.

`LocalSetting` =2,4: The control outputs are tracked to the feedbacks.

All three feedbacks are available and plausible means that always only one feedback is active. The corresponding control output is tracked:

```
Spd1 = FbkSpd1,
Spd2 = FbkSpd2,
StopCtrl = FbkStop.
```

When all feedbacks are false then the drive is between of two positions. In this case no monitoring is started. When two feedbacks are active at the same time, then they are not plausible. In this case the static monitoring counter starts. In both cases "all feedbacks are reset" or "not plausible feedbacks" with `Feature2Bit21` =1 all the control outputs are set to false (No command state) else control outputs will be hold.

If two feedbacks are available then they are always plausible besides they are both active.

| NoFbkStop =1 (default) | NoFbkSp1 =1 | NoFbkSp2 =1 |
|---|---|---|
| `Spd1 = FbkSpd1,`<br>`Spd2 = FbkSpd2,`<br>`StopCtrl = NOT FbkSpd1 AND NOT FbkSpd2` | `Spd1 = NOT FbkStop AND NOT FbkSpd2`<br>`Spd2 = FbkSpd2,`<br>`StopCtrl = FbkStop` | `Spd1 = FbkSpd1,`<br>`Spd2 = NOT FbkStop AND NOT FbkSpd1`<br>`StopCtrl = FbkStop` |

When they are not plausible the control outputs will be hold.

When only one feedback is available the concerned control output is tracked. The other control outputs are hold if the available feedback is false.

| NoFbkStop =0 | NoFbkSp1 =0 | NoFbkSp2 =0 |
|---|---|---|
| `StopCtrl = FbkStop,` `Spd1,Spd2` are hold if `FbkStop =0` | `Spd1= FbkSpd1,` `StopCtrl,Spd2` are hold if `FbkSpd1=0` | `Spd2 = FbkSpd2,` `StopCtrl, Spd1` are hold if `FbkSpd2=0` |

When all feedbacks are not available the control outputs will be hold.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and Bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 82) section.

Motor actions you can control in auto mode:

- starting with speed 1 (`Spd1Aut` = 1)
- starting with speed 2 (`Spd2Aut` = 1)
- stopping (`StopAut` = 1).

### "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the section Manual and automatic mode for motors, valves and dosers (Page 82).

Motor actions you can control in "manual mode":

- starting with speed 1 (`Spd1Man` = 1)
- starting with speed 2 (`Spd2Man` = 1)
- stopping (`StopMan` = 1).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature2.Bit1` =0 and `Feature2.Bit1` =1 is as shown below:

| Output Parameters | **`Feature2.Bit1` =0** | **`Feature2.Bit1` =1** |
|---|---|---|
| FbkSpd1Out.ST | 16#80 | 16#60 |
| FbkSpd2Out.ST | 16#80 | 16#60 |
| RunSpd1.ST | 16#80 | 16#60 |
| RunSpd2.ST | 16#80 | 16#60 |
| Stop.ST | 16#80 | 16#60 |
| Starting.ST | 16#80 | 16#60 |

| Output Parameters | Feature2.Bit1 =0 | Feature2.Bit1 =1 |
|---|---|---|
| Stoping.ST | 16#80 | 16#60 |
| AV_Out.ST | 16#80 | 16#60 |

### See also

MotSpdL block diagram (Page 1590)

MotSpdL I/Os (Page 1578)

MotSpdL messaging (Page 1577)

MotSpdL error handling (Page 1574)

MotSpdL functions (Page 1563)

Description of MotSpdL (Page 1556)

## 8.6.3 MotSpdL functions

### Functions of MotSpdL

The functions for this block are listed below.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" |
| 1 | 1 = Operator can switch to "manual mode" |
| 2 | 1 = Operator can switch to "local mode" |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can stop the motor |
| 5 | 1 = Operator can start the motor at speed 1 |
| 6 | 1 = Operator can start the motor at speed 2 |
| 7 | 1 = Operator can reset the motor |
| 8 | 1 = Operator can define or change the monitoring time for startup |
| 9 | 1 = Operator can define the monitoring time for the status |
| 10 | 1 = Operator can activate the monitoring time function (Bit 8 - 9) |
| 11 | 1 = Operator can activate the Simulation function |
| 12 | 1 = Operator can activate the Release for maintenance function |

| Bit | Function |
|---|---|
| 13 | 1 = Operator can change the limit (AV) for high alarm |
| 14 | 1 = Operator can change the limit (AV) for high warning |
| 15 | 1 = Operator can change the limit (AV) for high tolerance |
| 16 | 1 = Operator can change the limit (AV) for hysteresis |
| 17 | 1 = Operator can change the limit (AV) for low alarm |
| 18 | 1 = Operator can change the limit (AV) for low warning |
| 19 | 1 = Operator can change the limit (AV) for low tolerance |
| 20 | 1 = Operator can activate / deactivate messages via AV_AH_MsgEn |
| 21 | 1 = Operator can activate / deactivate messages via AV_WH_MsgEn |
| 22 | 1 = Operator can activate / deactivate messages via AV_TH_MsgEn |
| 23 | 1 = Operator can activate / deactivate messages via AV_TL_MsgEn |
| 24 | 1 = Operator can activate / deactivate messages via AV_WL_MsgEn |
| 25 | 1 = Operator can activate / deactivate messages via AV_AL_MsgEn |
| 26 | 1 = Operator can change the simulation value SimAV |
| 27 - 29 | Not used |
| 30 | 1 = Operator can define the monitoring time for stopping |
| 31 | 1 = Operator can execute the rapid stop |

The block has the following permissions for the OS1Perm parameter:

| Bit | Function |
|---|---|
| 0 - 15 | reserved |

**Note**

If you interconnect a parameter that is also listed in OS_Perm as a parameter, you have to reset the corresponding OS_Perm bit.

### Restart lock after switching off the motor

After switching off or stopping the motor, it can only be restarted after the time set with the IdleTime input parameter. When the "Stop" command is given, the motor goes immediately into "Stop" mode, and IdleTime starts after the feedback (FbkSpd1 and FbkSpd2 = 0) is given. The motor cannot be started again until the IdleTime has expired.

The IdleTime parameter can be set independently of the MonTiDynamic parameter.

### Limit monitoring of an additional analog value

This block provides the standard function Limit monitoring of an additional analog value (Page 98).

### Limit monitoring with hysteresis

This block provides the standard function Limit monitoring with hysteresis (Page 104).

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Interlocks

This block provides the following interlocks:

`Feature2.Bit16` = 0:

- Activation enable
- Interlock without reset ("Interlock")
- Interlock with reset ("Protection")
- Permission for stop ("Permission stop")

`Feature2.Bit16` = 1:

- Activation enable speed1
- Activation enable speed2
- Interlock speed1 without reset ("Interlock speed1")
- Interlock speed2 without reset ("Interlock speed2")
- Interlock open with reset ("Protection interlock speed1")
- Interlock close with reset ("Protection interlock speed2")

Refer to the section Interlocks (Page 107) as well as Influence of the signal status on the interlock (Page 112).

### Motor protection function

This block provides the standard function Motor protection function (Page 106).

### Rapid stop for motors

This block provides the standard function Rapid stop for motors (Page 115).

### Disabling interlocks

This block provides the standard function Disabling interlocks (Page 111).

### Resetting the block in case of interlocks or errors

This block provides the standard function Resetting the block in case of interlocks or errors (Page 47).

### External error (FaultExt), external control system fault (CSF)

This block provides the possibility to pass an external error via the FaultExt parameter or an external control system fault via the CSF parameter. See MotSpdL error handling (Page 1574).

### Group error

This block provides the standard function Outputting group errors (Page 130).

The following parameters are taken into consideration when forming the group error:

- CSF
- Trip
- MonDynErr
- MonStaErr
- FaultExt

### Outputting a signal for start readiness

This block provides the standard function Outputting a signal for start readiness (Page 59).

### Forming the group status for interlocks

This block provides the standard function Forming the group status for interlock information (Page 113).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status ST_Worst for the block is formed from the following parameters:

- FbkSpd1Out.ST
- FbkSpd2Out.ST
- LocalLi.ST
- Spd1Local.ST
- StopLocal.ST
- Spd2Local.ST
- Trip.ST
- AV_Out.ST
- Spd1Chn.ST
- Spd2Chn.ST
- Spd1Aut.ST (only if Feature2.Bit10 = 1)

- `Spd2Aut.ST` (only if `Feature2.Bit10` = 1)
- `StopAut.ST` (only if `Feature2.Bit10` = 1)

### Considering bad quality of automatic commands or external values

If the `Feature2` bit Considering bad quality of automatic commands or external values (Page 200) is set to 1 and one of the following parameters has bad signal status (16#00 or 16#28), the block goes to the neutral position (motor stop) in the "Automatic" mode:

- `Spd1Aut.ST`
- `Spd2Aut.ST`
- `StopAut.ST`

### Forcing operating modes

This block provides the standard function Forcing operating modes (Page 45).

The following states can be enforced:

- Speed 1 (`Spd1Force`)
- Speed 2 (`Spd2Force`)
- Stop (`StopForce`)

### Feedback monitoring

This block provides the standard function Monitoring the feedbacks (Page 104).

### Disabling feedback

This block provides the standard function for Disabling feedback for valve and motors (Page 106).

Feedback monitoring can be deactivated separately for each feedback with `NoFbkSpd1` or `NoFbkSpd2` or `NoFbkStop` as required.

The output signals of feedback are tracked by the control and the other available feedbacks:

| NoFbkSpd1 | NoFbkSpd2 | NoFbkStop | Comment |
|---|---|---|---|
| 0 | 0 | 1 | **Default:**<br>`FbkSpd1Out = FbkSpd1;`<br>`FbkSpd2Out = FbkSpd2;`<br>`FbkStopOut = StopCtrl AND NOT FbkSpd1 AND NOT FbkSpd2;`<br>**Local mode with Local-Setting =2/4:**<br>`FbkSpd1Out = FbkSpd1;`<br>`FbkSpd2Out = FbkSpd2;`<br>`FbkStopOut = NOT FbkSpd1 AND NOT FbkSpd2;` |
| 1 | 0 | 1 | `FbkSpd1Out = Spd1 AND NOT FbkSpd2;`<br>`FbkSpd2Out = FbkSpd2;`<br>`FbkStopOut = StopCtrl AND NOT FbkSpd2;` |
| 0 | 1 | 1 | `FbkSpd1Out = FbkSpd1;`<br>`FbkSpd2Out = Spd2 AND NOT FbkSpd1;`<br>`FbkStopOut = StopCtrl AND NOT FbkSpd1;` |
| 1 | 1 | 1 | `FbkSpd1Out = Spd1;`<br>`FbkSpd2Out = Spd2;`<br>`FbkStopOut = StopCtrl;` |

| NoFbkSpd1 | NoFbkSpd2 | NoFbkStop | Comment |
|---|---|---|---|
| 1 | 0 | 0 | `FbkFwdOut = Spd1 AND NOT FbkSpd2 AND NOT FbkStop;`<br>`FbkSpd2Out = FbkSpd2;`<br>`FbkStopOut = FbkStop;`<br>**Local mode with Local-Setting =2/4:**<br>`FbkSpd1Out = NOT FbkSpd2 AND NOT FbkStop;`<br>`FbkSpd2Out = FbkSpd2;`<br>`FbkStopOut = FbkStop;` |
| 0 | 1 | 0 | `FbkSpd1Out = FbkSpd1;`<br>`FbkSpd2Out = Spd2 AND NOT FbkSpd1 AND NOT FbkStop;`<br>`FbkStopOut = FbkStop;`<br>**Local mode with Local-Setting =2/4:**<br>`FbkSpd1Out = FbkSpd1;`<br>`FbkSpd2Out = NOT FbkSpd1 AND NOT FbkStop;`<br>`FbkStopOut = FbkStop;` |
| 0 | 0 | 0 | `FbkSpd1Out = FbkSpd1;`<br>`FbkSpd2Out = FbkSpd2;`<br>`FbkStopOut = FbkStop;` |
| 1 | 1 | 0 | `FbkSpd1Out = Spd1 AND NOT FbkStop;`<br>`FbkSpd2Out = Spd2 AND NOT FbkStop;`<br>`FbkStopOut = FbkStop;` |

### Using StopCtrl with mode "No Cmd"

This block provides the standard function Control with output StopCtrl and state “No Cmd” (Page 214)

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Specify warning times for control functions

This block provides the standard function Specifying warning times for control functions at motors and valves (Page 56).

You can generate warning signals when, for example, motors are started. Warning signals can be generated in the following modes:

- Manual mode (Page 82) (`WarnTiMan` input parameter)
- Automatic mode (Page 82) (`WarnTiAut` input parameter)

You specify the warning times in seconds using the input parameters `WarnTiMan` and `WarnTiAut`. If, for example, a motor is started, then this is displayed at the output parameter with `WarnAct` = 1. The motor then starts after the set warning time has expired and `WarnAct` then goes back to 0.

A corresponding warning is not output if the warning times (`WarnTiMan` or `WarnTiAut`) are specified with a smaller value than the `SampleTime` parameter.

### Step control mode for the speed change

Use the input parameter `SwiOverTi` and the `Feature.Bit5` (Specifying switching mode (Page 178)) to specify how the motor is to carry out the speed change.

In so doing use the input parameter `SwiOverTi` to adjust the switching time. You have the following options:

- Switching on and off occurs immediately
- Switching on via speed 1
- Switching off via speed 1

**Switching on and off occurs immediately:** This setting enables you to switch directly from the "Off" state to speed 2 or from speed 2 (`Spd2`) to "Off".

**Switching on via speed 1:** The speed change from the "Off" condition to speed 2 (`Spd2`) is accomplished via speed 1 (`Spd1`) and after expiration of the time at the parameter `SwiOverTi`.

**Switching off via speed 1:** The speed change from speed 2 (`Spd2`) to the "Off" state is accomplished via speed 1 (`Spd1`) and after expiration of the time at parameter `SwiOverTi`.

| Step control mode | **SwiOverTi** | **Feature.Bit5** |
|---|---|---|
| Switching on and off occurs immediately | = 0.0 | 0/1 |
| Switching on only via speed 1 | > 0.0 | 0 |
| Switching on and off via speed 1 | > 0.0 | 1 |

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Additional value (`SimAV`, `SimAV_Li`)

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

### Neutral position

This block provides the standard function Neutral position for motors, valves and controllers (Page 54).

### Output signal as a pulse signal or static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 57).

### Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 230).

### Resynchronization

This block provides the standard function Resynchronization (Page 258).

Resynchronization is switched on over feature bits:

| | |
|---|---|
| Feature2 Bit17 =1 | Resynchronization after CPU startup |
| Feature2 Bit18 =1 | Resynchronization after Simulation or after Out of service |
| Feature2 Bit19 =1 | Resynchronization after control system fault |
| Feature2 Bit20 =1 | Resynchronization with 0->1 trigger at StateTrkOn input |

If there are no restrictions the controls are tracked as follow:

**NoFbkSpd1 =0, NoFbkSpd2 =0, NoFbkStop =1 (default):**

Spd1 := FbkSpd1;

Spd2 := FbkSpd2;

StopCtrl := NOT FbkSpd1 AND NOT FbkSpd2;

In case if there are no plausible feedback positions (FbkSpd1 =1, FbkSpd2 =1) the controls are not changed.

**NoFbkSpd1 =0, NoFbkSpd2 =0, NoFbkStop =0:**

Plausible feedback positions (only one feedback signal is active) are tracked as control position.

Spd1 := FbkSpd1;

Spd2 := FbkSpd2;

StopCtrl := FbkStop;

In case if there are no plausible feedback positions (two or three feedback signals are active) the control position depends on Feature2 Bit21.

- Feature2 Bit21 =0 (default): Control position is not changed.
- Feature2 Bit21 =1: Control position becomes "No Cmd" Spd1 := 0; Spd2 := 0; StopCtrl := 0;

**NoFbkSpd1 =0, NoFbkSpd2 =1, NoFbkStop =0:**

Spd1 := FbkSpd1 AND NOT FbkStop;

Spd2 := NOT FbkSpd1 AND NOT FbkStop;

StopCtrl := FbkStop;

In case if there are no plausible feedback positions (FbkSpd1 =1, FbkStop =1) the controls are not changed.

**NoFbkFwd =1, NoFbkRev =0, NoFbkStop =0:**

Spd1 := NOT FbkSpd2 AND NOT FbkStop;

Spd2 := FbkSpd2 AND NOT FbkStop;

StopCtrl := FbkStop;

In case if there are no plausible feedback positions (FbkSpd2 =1, FbkStop =1) the controls are not changed.

With other combinations of **NoFbkSpd1**, **NoFbkSpd2**, **NoFbkStop** the controls are not changed.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the section Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 3 | Enabling resetting of commands for the control settings (Page 172) |
| 4 | Setting switch or button mode (Page 177) |
| 5 | Specifying switching mode (Page 178) |
| 9 | Resetting via input signals in the event of interlocking (Protection) or errors (Page 174) |
| 10 | Exiting local mode (Page 189) |
| 11 | Activating the run time of feedback signals (Page 164) |
| 13 | Separate monitoring time for stopping the motor (Page 180) |
| 14 | Enabling rapid stop via faceplate (Page 179) |
| 17 | Enabling bumpless switchover to automatic mode for valves, motors, and dosers (Page 184) |
| 18 | Activating error state for external process control error CSF (Page 162) |
| 19 | Reset even with locked state (Page 176) |
| 20 | Disable calculation of impulse control in LocalSetting 2 and 4 (Page 200) |

| Bit | Function |
|---|---|
| 21 | Enable bumpless switchover to "Automatic" mode for operator only (Page 183) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 23 | Limits disabled in faceplate (Page 206) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 27 | Interlock display with LocalSetting 2 or 4 (Page 189) |
| 28 | Disabling operating points (Page 156) |
| 29 | Signaling limit violation (Page 181) |
| 30 | Set reset depending on the operating mode or the LiOp parameter (Page 174) |
| 31 | Activating reset of protection / error in manual mode (Page 175) |

In pushbutton operation (Bit 4 = 0) the automatic commands in "automatic" mode are latching, in other words `Spd1Aut`, `Spd2Aut`, `StopAut` can be reset to 0 after changing the control. In "manual" and "local" modes, however, the automatic commands are not saved and in the absence of automatic commands the automatic control is tracked.

In switching mode (Bit 4 = 1), control is selected with the static signals `Spd1Aut`, `Spd2Aut`. If `Spd1Aut`, `Spd2Aut` inputs are not set, the motor is stopped. Control via `StopAut` is not needed. If the "Activate command reset for control" function (Bit 3 = 1) is also enabled, the inputs `Spd1Aut`, `Spd2Aut` are reset to the neutral position after evaluation in the block.

### Configurable reactions using the `Feature2` parameter

| Bit | Function |
|---|---|
| 1 | Changes signal status of outputs in OOS (Page 208) |
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 3 | Control priority in the event of invalid or not plausible input commands (Page 186) |
| 4 | Setting switch or button mode for local commands (Page 195) |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) |
| 8 | Forcing commands in the Local mode (Page 198) |
| 9 | Button mode with triggered local control commands (Page 209) |
| 10 | Considering bad quality of automatic commands or external values (Page 200) |
| 11 | Suppress MsgLock and “Out of service” mode for a connected message block (Page 202) |
| 13 | Resetting the Pending Error in Local Mode 1 and 3 (Page 210) |
| 16 | Separate interlock for each direction, position, or speed (Page 180) |
| 17 | Resynchronization after startup (Page 213) |
| 18 | Resynchronization after out of service or after simulation (Page 213) |
| 19 | Resynchronization after CSF (Page 214) |
| 20 | Resynchronization with StateTrkOn input (Page 214) |
| 21 | Control with output StopCtrl and state “No Cmd” (Page 214) |

### Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 239).

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Button labels

This block provides the standard function Labeling of buttons and text (Page 235)

Instance-specific text can be configured for the following parameters:

- `Spd1Man`
- `Spd2Man`
- `StopMan`
- `RapidStp`

### See also

MotSpdL block diagram (Page 1590)

MotSpdL modes (Page 1561)

EventTs functions (Page 2046)

MotSpdL I/Os (Page 1578)

## 8.6.4 MotSpdL error handling

### Error handling of MotSpdL

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Mode switchover error
- Invalid or not plausible input commands
- Control system fault (CSF)

### Overview of error numbers

The ErrorNum I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. |
| 41 | The value for the LocalSetting I/O is not within the approved limit of 0 to 4. |
| 42 | LocalSetting = 0 or LocalSetting = 3 or LocalSetting = 4 and LocalLi = 1 |
| 51 | Spd1Local = 1 and StopLocal = 1<br>Spd2Local = 1 and StopLocal = 1<br>Spd1Local = 1 and Spd2Local = 1<br>Spd1Aut = 1 and StopAut = 1<br>Spd2Aut = 1 and StopAut = 1<br>Spd1Aut = 1 and Spd2Aut = 1<br>AutModLi = 1 and ManModLi = 1<br>Spd1Force = 1 and StopForce = 1<br>Spd2Force = 1 and StopForce = 1<br>Spd1Force = 1 and Spd2Force = 1 |
| 52 | LocalAct = 1 and LocalSetting = 2 or 4 and SimOn = 1 |

### Mode switchover error

This error can be output by the block, see the section Error handling (Page 128).

### Invalid or not plausible input commands

This error can be output by the block, see the section Error handling (Page 128).

For the following invalid or not plausible input commands, the control output can be kept or switched to the neutral position. This depends on the function Control priority in the event of invalid or not plausible input commands (Page 186).

| **Operating mode** | **Invalid or not plausible input commands** | **Control reaction with** `Feature2.Bit3` =1 |
|---|---|---|
| Local: `Localsetting` = 1 or `Localsetting` = 3 | Pushbutton operation for local mode (`Feature2.Bit4` = 0):<br>`Spd1Local` = 1 and `Spd2Local` = 1 or<br>`Spd1Local` = 1 and `StopLocal` = 1 or<br>`StopLocal` = 1 and `Spd2Local` = 1<br>Switching mode (`Feature2.Bit4` = 1):<br>`Spd1Local` = 1 and `Spd2Local` = 1 | Motor is stopped |
| Local: `Localsetting` = 1 or `Localsetting` = 3 and forcing | `Spd1Force` = 1 and `Spd2Force` = 1 or<br>`Spd1Force` = 1 and `StopForce` = 1 or<br>`StopForce` = 1 and `Spd2Force` = 1 | |
| Forcing and no "local mode" | `Spd1Force` = 1 and `Spd2Force` = 1 or<br>`Spd1Force` = 1 and `StopForce` = 1 or<br>`StopForce` = 1 and `Spd2Force` = 1 | |
| "Automatic mode" and no forcing | Pushbutton operation (`Feature.Bit4` = 0):<br>`Spd1Aut` = 1 and `Spd2Aut` = 1 or<br>`Spd1Aut` = 1 and `StopAut` = 1 or<br>`StopAut` = 1 and `Spd2Aut` = 1<br>Switching mode (`Feature.Bit4` = 1):<br>`Spd1Aut` = 1 and `Spd2Aut` = 1 | |
| "Manual mode" and no forcing | `Spd1Man` = 1 and `Spd2Man` = 1 or<br>`Spd1Man` = 1 and `StopMan` = 1 or<br>`StopMan` = 1 and `Spd2Man` = 1 | |

## Control system fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 128) section for more on this.

## See also

MotSpdL block diagram (Page 1590)

MotSpdL I/Os (Page 1578)

MotSpdL functions (Page 1563)

MotSpdL modes (Page 1561)

Description of MotSpdL (Page 1556)

MotSpdL messaging (Page 1577)

## 8.6.5 MotSpdL messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Instance-specific messages

### Process control fault

The following control system error messages can be output:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 1 | AS process control message - fault | $$BlockComment$$<br>Motor feedback error |
| | SIG 2 | AS process control message - fault | $$BlockComment$$<br>Motor protection triggered |
| | SIG 3 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If it changes to CSF = 1, a process control fault is triggered (MsgEvId1, SIG 3).

### Instance-specific messages

You can use up to three instance-specific messages with this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External message 1 |
| | SIG 5 | AS process control message - fault | $$BlockComment$$<br>External message 2 |
| | SIG 6 | AS process control message - fault | $$BlockComment$$<br>External message 3 |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance `MsgEvId1`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | `ExtVa104` |
| 5 | `ExtVa105` |
| 6 | `ExtVa106` |
| 7 | `ExtVa107` |
| 8 | `ExtVa108` |
| 9 | Reserved |
| 10 | Reserved |

The associated values 4 ... 8 are allocated to the parameters `ExtVa104` ... `ExtVa108` and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

MotSpdL block diagram (Page 1590)

MotSpdL modes (Page 1561)

MotSpdL error handling (Page 1574)

## 8.6.6 MotSpdL I/Os

I/Os of MotSpdL

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

Table 8-1

| Parameter | Description | Type | Default |
|---|---|---|---|
| `AutModLi*` | 1 = "Automatic mode" via interconnection or SFC (controlled by `ModLiOp` = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `AutModOp*` | 1 = "Automatic mode" via operator (controlled by `ModLiOp` = 0) | BOOL | 0 |
| `AV` | Input additional analog value, to be connected to `AV_Tech` of the AV block | ANY | |
| `AV_AH_Lim` | Limit high alarm | REAL | 95.0 |
| `AV_AL_Lim` | Limit low alarm | REAL | 5.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| AV_Hyst | Hysteresis for alarm, warning and tolerance limits | REAL | 1.0 |
| AV_TH_Lim | Limit high tolerance | REAL | 85.0 |
| AV_TL_Lim | Limit low tolerance | REAL | 15.0 |
| AV_WH_Lim | Limit high warning | REAL | 90.0 |
| AV_WL_Lim | Limit low warning | REAL | 10.0 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkTypeData* | This input can not be edit or linked and is for internal usage only! | | |
| BypProt | 1 = Bypassing interlock in "local mode" and in "simulation" | BOOL | 0 |
| CSF | 1 = External error (control system faultError handling (Page 128)) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| EventTsIn | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg3 | Binary input for freely selectable message 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa104 | Associated value 4 for messages (MsgEvID1) | ANY | |
| ExtVa105 | Associated value 5 for messages (MsgEvID1) | ANY | |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| ExtVa108 | Associated value 8 for messages (MsgEvID1) | ANY | |
| FaultExt | 1 = External error<br>Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkSpd1 | 1= Motor feedback running Speed1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| FbkSpd2 | 1= Motor feedback running Speed2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| FbkStop | 1= Motor feedback stoped | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| Feature | I/O for additional functions (Page 1563)<br>1 = Separate monitoring time for stopping the motor | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 13: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0<br>• 0<br>• 0 |
| Feature2 | I/O for additional functions (Page 1563) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| IdleTime* | Wait time for restart of motor in [s] | REAL | 5.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Intlock | 0 = Interlock / Interlock open is activated.<br>Once the interlock condition has cleared, the block can be operated without reset.<br>1 = Interlock / Interlock open is deactivated. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| Intl_En | 1 = Parameter Intlock is enabled. | BOOL | 1 |
| IntlSp2 | 0 = Interlock close is activated.<br>Once the interlock condition has cleared, the block can be operated without reset.<br>1 = Interlock close is deactivated. | BOOL | 1 |
| IntlSp2En | 1 = Parameter IntlSp2 is enabled. | BOOL | 1 |
| LocalLi | 1 = Activate "local mode" via plant signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalOp* | 1 = "Local mode" via operator | BOOL | 0 |
| LocalSetting | Properties for the Local mode (Page 86) | INT | 0 |
| ManModLi* | 1 = "Manual mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManModOp* | 1 = "Manual mode" via OS operator (controlled by ModLiOp = 0) | BOOL | 1 |
| ModLiOp | Switchover of operating mode between:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Monitor | 1 = Feedback monitoring | BOOL | 1 |
| MonSafePos | 1 = Go to neutral position in the event of monitoring errors  Motors: Stop is the neutral position | BOOL | 1 |
| MonTiDynamic* | Monitoring time for feedback errors  or feedback start error after successful operation in [s] | REAL | 3.0 |
| MonTiDyStop* | Monitoring time for feedback stop errors after successful operation in [s] | REAL | 3.0 |
| MonTiStatic* | Monitoring time for feedback errors without operation in [s] | REAL | 3.0 |
| MS_RelOp | 1= Release for maintenance via OS operator | BOOL | 0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_RelOp* | 1 = Release for maintenance via OS operator | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| NoFbkSp1 | 1= Motor Feedback running Speed1 not present | BOOL | 0 |
| NoFbkSp2 | 1= Motor Feedback running Speed2 not present | BOOL | 0 |
| NoFbkStop | 1= Motor Feedback stopped not present | BOOL | 1 |
| `Occupied` | 1 = In use by a batch | BOOL | 0 |
| `OosLi` | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `OosOp*` | 1 = "Out of service", via OS operator | BOOL | 0 |
| `OpSt_In` | Input parameter for local operator authorization, connected with the `Out` output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| `OS_Perm` | I/O for operator permissions (Page 1563)<br>1 = Operator can set or change the monitoring time for "Control: Start"<br>1 = Operator can set the monitoring time for "Control:Stop" | STRUCT<br>• Bit 0: BOOL<br>• Bit 8: BOOL<br>• Bit 20: BOOL<br>• Bit 30: BOOL | -<br>• 1<br>• 1<br>• 1<br>• 1 |
| `OS1Perm` | I/O for operator permissions (Page 1563) | STRUCT<br>• Bit 0: BOOL<br>• Bit 15: BOOL | -<br>• 1<br>• 1 |
| `Permit` | 1 = Activation enable / Activation open enable.<br>0 = No activation enable / No activation open enable. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| PermitSafePos | 1=Motor stop is allowed | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| PermSf_En | 1=Permit stop enabled | BOOL | 1 |
| `Perm_En` | 1 = Parameter `Permit` enable | BOOL | 1 |
| `PermSp2` | 1 = Activation Close enable<br>0 = No activation Close enable | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| `PermSp2En` | 1 = Parameter `PermSp2` is enabled | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Protect | 0 = Protective interlock / Protective interlock open is activated.<br>Once the interlock condition has cleared, the block must be reset.<br>1 = Protective interlock / Protective interlock open is deactivated. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| Prot_En | 1 = Parameter Protect enable | BOOL | 1 |
| ProtSp2 | 0 = Protective interlock close is activated.<br>Once the interlock condition has cleared, the block must be reset.<br>1 = Protective interlock close is deactivated. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| ProtSp2En | 1 = Parameter ProtSp2 enable | BOOL | 1 |
| PulseWidth* | Pulse width of control signal [s] | REAL | 3.0 |
| RapidStp* | Rapid stop for the motor:<br>0 = Motor On<br>1 = Motor Off | BOOL | 0 |
| Spd1Aut* | 1 = Activation of motor speed 1 in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Spd2Aut* | 1 = Activation of motor speed 2 in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Spd1ChnST | Signal status of output channel for Fwd<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| Spd2ChnST | Signal status of output channel for Rev<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| Spd1Force | 1 = Force activation of motor speed 1 in "manual mode" or in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Spd2Force | 1 = Force activation of motor speed 2 in "manual mode" or in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Spd1Local | 1 = Activation of motor speed 1 in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Spd2Local | 1 = Activation of motor speed 2 in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Spd1Man* | 1 = Activation of motor speed 1 in "manual mode" | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Spd2Man* | 1 = Activation of motor speed 2 in "manual mode" | BOOL | 0 |
| RstLi* | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | 1 = Reset via operator | BOOL | 0 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SimAV* | Additional value used for SimOn = 1 | REAL | 0.0 |
| SimAV_Li | Additional analog value that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | |
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | |
| StepNo | Batch step number | DWORD | 16#00000000 |
| StopAut* | 1 = Stopping the motor in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopForce | 1 = Force motor stop | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopLocal | 1 = Stopping the motor in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopMan* | 1 = Stopping the motor in "manual mode" | BOOL | 0 |
| SwOverTi* | Time for speed change | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Trip | 1 = Motor is in "good" state | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| UserAna1 | Analog auxiliary value 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UA1unit | Unit of measure for analog auxiliary value 1 | INT | 0 |
| UserAna2 | Analog auxiliary value 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UA1unit | Unit of measure for analog auxiliary value 1 | INT | 0 |
| UA2unit | Unit of measure for analog auxiliary value 2 | INT | 0 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |
| WarnTiAut* | Prewarning of motor start in "automatic mode" in [s] | REAL | 0.0 |
| StateTrkOn | Initiate tracking of state | BOOL | 0 |
| WarnTiMan* | Prewarning of motor start in "manual mode" in [s] | REAL | 0.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutAct | 1 = "Automatic mode" enabled<br>0 = "Manual mode" is enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutoST | 1 = Worst signal status of automatic commands | BYTE | 16#80 |
| AV_OpScale | Limit for scale in AV bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| AV_Out | Output additional analog value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| AV_Unit | Unit of measure for additional analog value | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| `CSF_Out` | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `CurrMon` | Remaining monitoring time [s] | DINT | 0 |
| `ENO` | 1 = Block algorithm completed without errors | BOOL | 0 |
| `ErrorNum` | Output of pending error number. For error numbers that can be output by this block, see MotSpdL error handling (Page 1574) | INT | -1 |
| `FaultExtOut` | FaultExt as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `FbkSpd1Out` | 1= Motor feedback running Speed1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `FbkSpd2Out` | 1= Motor feedback running Speed1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkStopOut | 1= Motor feedback stopped | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `GrpErr` | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `IntlockOut` | Intlock as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `IntlSp2Out` | IntlSp2 as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `LocalAct` | 1 = "Local mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `LockAct` | 1 = Interlock (`Intlock`, `Permit`, `Protect`) or `Trip` is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `ManAct` | 1 = "Manual mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MonDynErr | 1 = Feedback error or feedback start error due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MonDynStopErr | 1 = Feedback stop error due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MonStaErr | 1 = Feedback error due to unexpected feedback change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message acknowledgement status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | Alarm error 1 (output ERROR of first ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1Perm | WORD | 16#FFFF |
| OS1PermLog | Display of OS1Perm with settings changed by the block algorithm | WORD | 16#FFFF |
| P_Rst | 1= Pulse output for reset<br>The parameter persists for one cycle after a reset. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Spd1 | 1 = Pulse signal for starting the motor with speed 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Spd2 | 1 = Pulse signal for starting the motor with speed 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Stop | 0 = Pulse signal for stopping the motor | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PermitOut | Permit as Output | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| PermitSafePosOut | PermitSafePos as output | STRUCT | - |
| | | • Value: BOOL | • 1 |
| | | • ST: BYTE | • 16#FF |
| PermSp2Out | PermSp2 as Output | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| ProtectOut | Protect as Output | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| ProtSp2Out | ProtSp2 as Output | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| R_StpAct | 1 = Rapid stop of the motor is active | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| RdyToReset | 1 = Ready for reset via RstLi input or commands in "local", "automatic", or "manual mode" | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| RdyToStart | 1 = Active start readiness | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| RunSpd1 | 1 = Motor running at speed 1 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| RunSpd2 | 1 = Motor running at speed 2 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| SimAct | 1 = Simulation Active | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Spd1 | 1 = Control of motor: Speed 1 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| Spd2 | 1 = Control of motor: Speed 2 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Starting | 1 = Motor will start | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Status1 | Status word 1 (Page 1556) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1556) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1556) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 1556) | DWORD | 16#00000000 |
| Status5 | Status word 5 (Page 1556) | DWORD | 16#00000000 |
| Stop | 1 = Motor is stopping | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopCtrl | Control output to Stop the motor (go to Safe position) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Stoping | 1 = Motor will stop | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| TripOut | Trip as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| UA1unitOut | UA1 as output | INT | 0 |
| UA2unitOut | UA2 as output | INT | 0 |
| UserAna1Out | UserAna1 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| UserAna2Out | UserAna2 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| ResyncST | Worst Signal Status of inputs for resynchronization | BYTE | 16#80 |
| WarnAct | 1 = Prewarning for motor start active (parameters WarnTiAut and WarnTiMan) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

## See also

MotSpdL messaging (Page 1577)

MotSpdL block diagram (Page 1590)

MotSpdL modes (Page 1561)

Error handling (Page 128)

## 8.6.7 MotSpdL block diagram

### MotSpdL block diagram

A block diagram is not provided for this block.

### See also

MotSpdL I/Os (Page 1578)

MotSpdL messaging (Page 1577)

MotSpdL error handling (Page 1574)

MotSpdL functions (Page 1563)

MotSpdL modes (Page 1561)

Description of MotSpdL (Page 1556)

## 8.6.8 Operator control and monitoring

### 8.6.8.1 MotSpdL views

### Views of the MotSpdL block

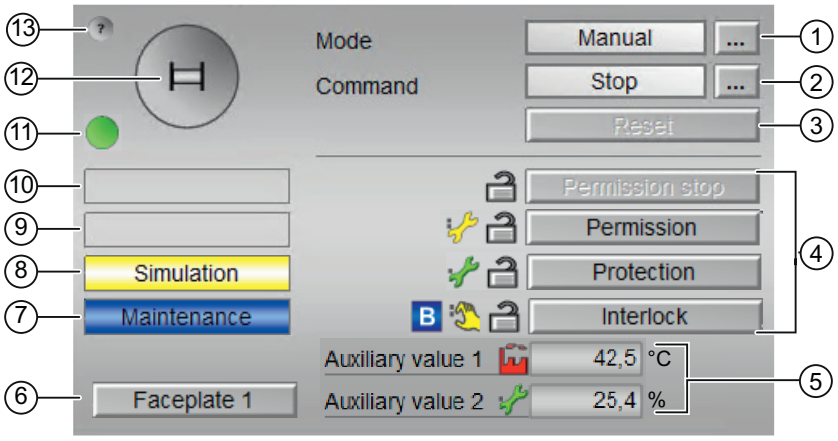
The block MotSpdL provides the following views:

- MotSpdL standard view (Page 1591)
- Alarm view (Page 336)
- Limit view of motors (Page 328)
- Trend view (Page 341)
- Parameter view for motors and valves (Page 320)
- MotSpdL preview (Page 1595)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for MotSpdL (Page 1600)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 8.6.8.2 MotSpdL standard view

#### MotSpdL standard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual and automatic mode for motors, valves and dosers (Page 82)
- Manual and automatic mode for motors, valves and dosers (Page 82)
- Local mode (Page 86)
- Out of service (Page 77)

Refer to section Switching operating states and operating modes (Page 289) for information on switching the operating mode.

#### (2) Starting and stopping the motor

This area shows you the default operating state for the motor. The following states can be shown and executed here:

- "Start >"
- "Start >>"
- "Stop"
- "Rapid stop"

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

If the text is configured for these commands, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (3) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in the section Resetting the block in case of interlocks or errors (Page 47).

### (4) Operating range for the interlock functions of the block

This display is only visible when the corresponding block input is connected.

You can use this button to control the interlock functions of the block. You can find additional information on this in the section Interlocking functions (Page 107).

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 113)), for example:

- Signal status (see Forming and outputting the signal status for technologic blocks (Page 117)), for example:

- Bypass information (see Forming the group status for interlock information (Page 113)):

If the text is configured for the interlocks, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (5) Display of auxiliary values

This display is only visible when the corresponding block input is connected.

You can use this area to display two auxiliary values that have been configured in the engineering system (ES). You can find additional information on this in the section Displaying auxiliary values (Page 239).

### (6) Navigation button for switching to the standard view of any faceplate

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234).

### (7) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this is available in section Release for maintenance (Page 70) Display area for block states.

### (8) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"
- "Delay"

You will find more detailed information on this in the sections Simulating signals (Page 64) and Display of delay times (Page 288).

### (9) Display area for block states

This area provides additional information on the operating state of the block:

- "Motor protection"
- "External error"
- "Status error"
- "Control error"
- "Internal error"
- "Changeover error"
- "Invalid Signal"

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 104) , Error handling (Page 128) (section "Invalid input signals" and "Mode switchover error") and Motor protection function (Page 106).

### (10) Display area for block states

This area provides additional information on the operating state of the block:

- "Forced stop"
- "Forced start >"
- "Forced start >>"
- "Request 0/1": Reset is expected but reset is not possible from faceplate. For more details, refer:Reset of errors during Request 0/1 (Page 382)

You can find additional information on this in the section Forcing operating modes (Page 45).

### (11) Automatic preview

This display is only visible in "manual mode", in "local mode", or with a reset request in "automatic mode", when the current output signals are not identical to the control in "automatic mode".

The display shows what state the motor would assume if you switched from "manual" or "local" mode to "automatic mode", or performed a reset to "automatic mode".

### (12) Status display of the motor

The current status of the motor is graphically displayed here.

You can find more information about this in section Block icon for MotSpdL (Page 1600).

### (13) Mini preview for Resynchronization

Displays the mini preview for Resynchronizing the signal state.

The mini preview displays the following states:

- Spd1
- Spd2
- Stop
- No Cmd

**Note**

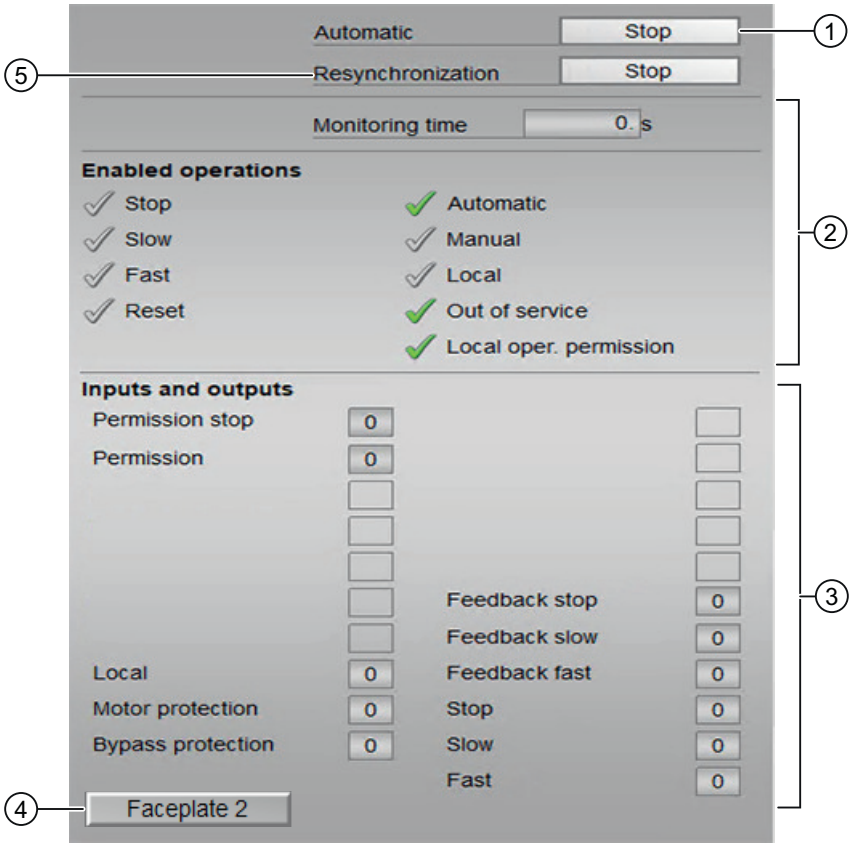
Mini preview for resynch is visible only if resynchronization command is different to control output value.

## See also

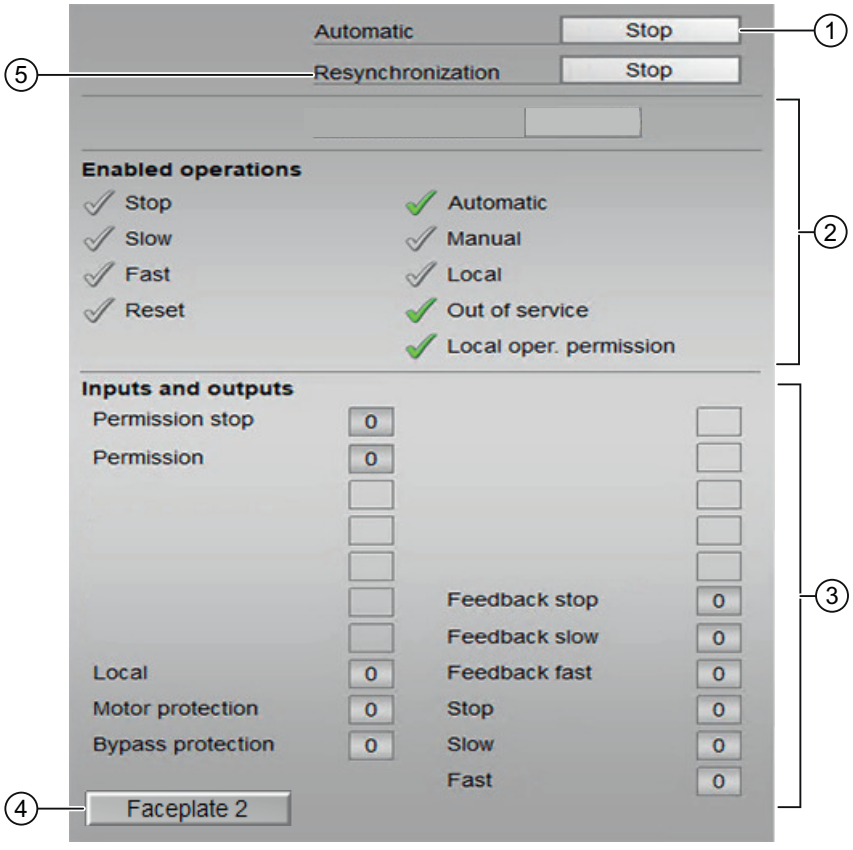
Functions of the blocks (Page 45)

### 8.6.8.3 MotSpdL preview

**Preview of MotSpdL**




Display of the current monitoring time is visible.




Display of the current monitoring time is not visible.

### (1) Automatic preview

This area shows you the block status after it has switched from the "Manual" mode or "Local" mode to the "Automatic" mode.

If the block is in "Automatic" mode, the current block state is displayed.

**Note**

- If the valve is closed and interlock locking is active (`SafePos`=0 Close), the automatic preview does not show the current status "Closed", instead it shows the active automatic command "Open".
- If the valve is in the end position and the automatic stop command is active, the automatic preview does not show the current status "Closed", but the active automatic command "Stop".

This area also displays the worst signal status of the following automatic commands:

- `Spd1Aut`
- `Spd2Aut`
- `StopAut`

### (2) Enable operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "Stop": You can stop the motor.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Slow": You can start the motor in the "slow" state.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Fast": You can start the motor in the "fast" state.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Reset": You can reset the motor after interlocks or errors.
- "Automatic": You can switch to "automatic mode".
- "Manual": You can switch to "manual mode".
- "Local": You can switch to "local mode".
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operator permission": Use the ← button to switch to the standard view of the OpStations block. For more information, see section Operator control permissions (Page 285).
- "Monitoring time": Display of the remaining monitoring time.

### (3) Displaying current control signals

This area shows the most important parameters for this block with the current selection:

- "Permission" (`Feature2.Bit16` **= 0**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Motor valve activation not enabled on OS
  - 1 = Enable for "slow"/"fast" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Permission slow" (`Feature2.Bit16` **= 1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Motor valve activation not enabled on OS
  - 1 = Enable for "slow" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Permission fast" (`Feature2.Bit16` **= 1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Motor valve activation not enabled on OS
  - 1 = Enable for "fast" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection" (`Feature2.Bit16` **= 0**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection slow" (`Feature2.Bit16` **= 1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).

- "Protection fast" (`Feature2.Bit16` **= 1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock" (`Feature2.Bit16` **= 0**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock slow" (`Feature2.Bit16` **= 1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock fast" (`Feature2.Bit16` **= 1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Local": 1 = Control signal for "Local mode" (LocalLi) is active
- "Motor protection": 1 = Motor is in "good" state
- "Interlock deact.":
  - 0 = Bypass disabled
  - 1 = Bypassing interlock in "local mode" and in "simulation"
- "Local stop": 1 = Stopping the motor in "local mode"
- "Local slow": 1 = Starting the motor in "local mode", slow
- "Local fast": 1 = Starting the motor in "local mode", fast
- "Feedback slow": 1 = Motor has started and is running slow
- "Feedback fast": 1 = Motor has started and is running fast
- "Feedback stop" : 1 = Motor has stopped

- "Stop": 1 = Stop motor
- "Slow": 1 = Motor is running slow
- "Fast": 1 = Motor is running fast
- "Channel Slow": Signal from the output channel block for "Slow"
- "Channel Fast": Signal from the output channel block for "Fast"
- "Permission stop": 1 = Motor can not be stopped in manual (by operator) or automatic (by logic) mode

**(4) Navigation button for switching to the standard view of any faceplate**

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

For more information, see section Opening additional faceplates (Page 234).

**(5) Resynchronization**

Displays the Resynchronization state and Resynchronization Signal status based on feedback value.

### 8.6.8.4 Block icon for MotSpdL

**Properties of the MotSpdLblock icon**

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as control system faults
- Operating modes
- Signal status, release for maintenance
- Displays for bypassing interlocks
- Interlocks
- Memo display
- Motor state display

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | "M" symbol with small speed display |
|  | 6 | "M" symbol with small speed display |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | |
|  | 6 | |
|  | 7 | |
|  | 8 | |

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 9 | |
|  | 10 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

## Motor state display

The following motor states are shown here:

| Symbol | Meaning |
|---|---|
|  | Motor started (motor symbol changes) |
|  | The motor is running |
|  | Motor stopped (motor symbol changes) |
|  | Motor idle |
|  | Error at motor (monitoring error, motor protection) |
|  | Motor out of service |
|  | “No Cmd” State |

# 8.7 ShrdResL - Multiplexer for shared resources (Large)

## 8.7.1 Description of ShrdResL

### Object name (type + number) and family

Type + number: FB 1917

Family: Drives

### Area of application for ShrdResL

The block is used for the following application:

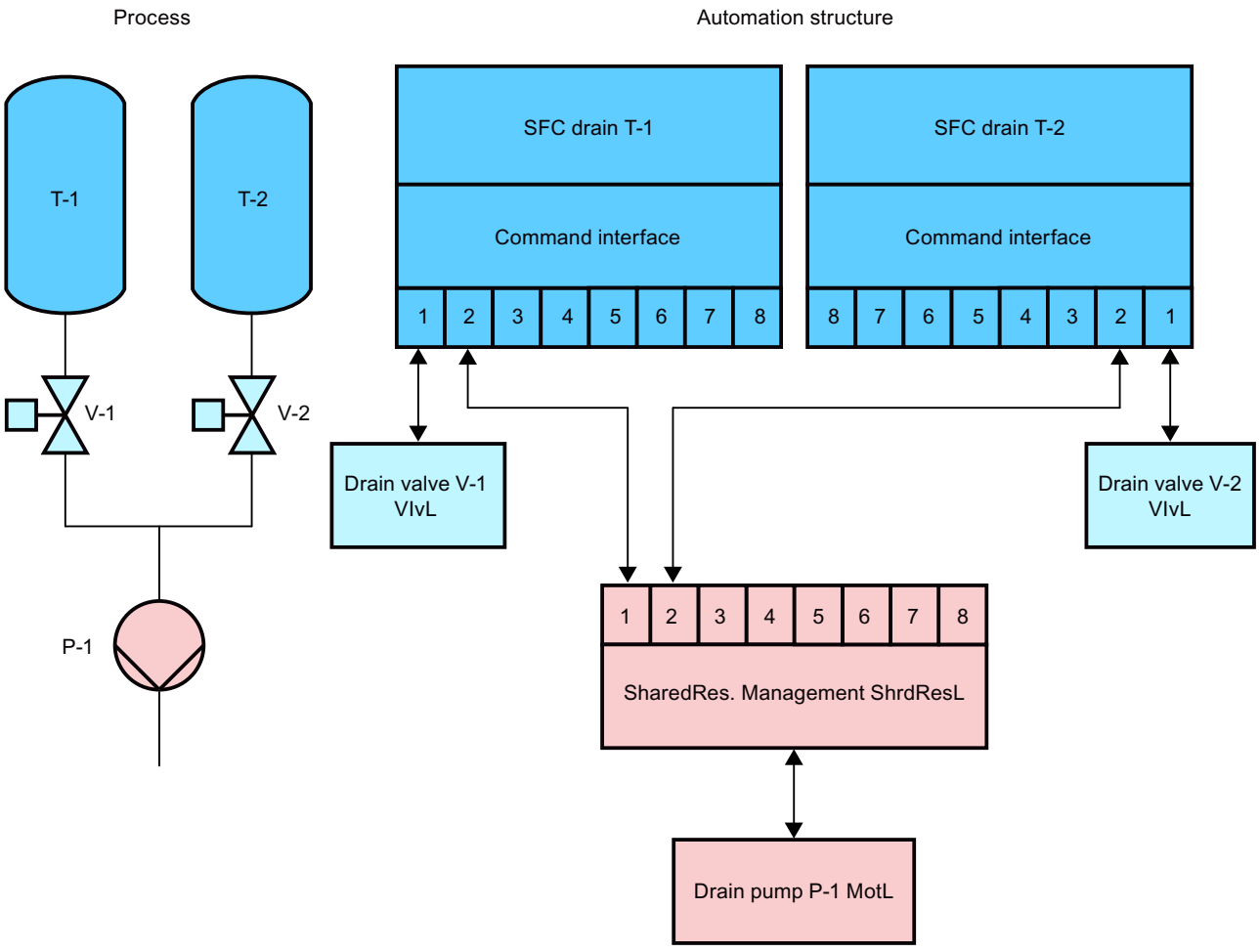
- Organize individual requests for downstream block from upstream applications.

### How it works

The block coordinates the access from up to eight SFC step sequencers to a technological block of the families "Drives" or "Dose".These blocks include:

- Motors
- Valves
- Dosers

The block has eight channels, each with a standardized command interface.




In case of multiple allocation requests, the channel to be allocated depends on the priority of each channel and a strategy mode which coordinates the sequence of the allocated channels.

As soon as a channel is allocated, its command interface is aligned in accordance with the command interface at the output.

**Note**

A cascading of channels is not possible.

## Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

## Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in the section ShrdResL I/Os (Page 1614).

| Status bit | Parameter |
|---|---|
| 0 | `ReadyIn`, Readiness signal of the interconnected technological block |
| 1 | `ChnEn_1` |
| 2 | `ChnEn_2` |
| 3 | `ChnEn_3` |
| 4 | `ChnEn_4` |
| 5 | `Ready`, Channel allocation is possible |
| 6 | `ActChnNo` = 1, 1 = Channel 1 is enabled |
| 7 | `ActChnNo` = 2, 1 = Channel 2 is enabled |
| 8 | `ActChnNo` = 3, 1 = Channel 3 is enabled |
| 9 | `ActChnNo` = 4, 1 = Channel 4 is enabled |
| 10 | `MultiOcc`, 1 = Request for more than one channel |
| 11 | Not used |
| 12 | `BatchEn` |
| 13 | `Occupied` |
| 14 - 16 | Not used |
| 17 | `NxtChnNo` = 1, 1 = Channel 1 is the next allocated channel |
| 18 | `NxtChnNo` = 2, 1 = Channel 2 is the next allocated channel |
| 19 | `NxtChnNo` = 3, 1 = Channel 3 is the next allocated channel |
| 20 | `NxtChnNo` = 4, 1 = Channel 4 is the next allocated channel |
| 21 | `BaEn_1` |
| 22 | `BaEn_2` |
| 23 | `BaEn_3` |
| 24 | `BaEn_4` |
| 25 | `Occ_1` |
| 26 | `Occ_2` |
| 27 | `Occ_3` |
| 28 | `Occ_4` |
| 29 - 30 | Not used |
| 31 | Internal error |

### Status word allocation for `Status2` parameter

You can find a description for each parameter in the section ShrdResL I/Os (Page 1614).

| Status bit | Parameter |
|---|---|
| 0 | Not used |
| 1 | `ChnEn_5` |
| 2 | `ChnEn_6` |
| 3 | `ChnEn_7` |
| 4 | `ChnEn_8` |

| Status bit | Parameter |
|---|---|
| 5 | Not used |
| 6 | `ActChnNo` = 5, 1 = Channel 5 is enabled |
| 7 | `ActChnNo` = 6, 1 = Channel 6 is enabled |
| 8 | `ActChnNo` = 7, 1 = Channel 7 is enabled |
| 9 | `ActChnNo` = 8, 1 = Channel 8 is enabled |
| 10 - 16 | Not used |
| 17 | `NxtChnNo` = 5, 1 = Channel 5 is the next allocated channel |
| 18 | `NxtChnNo` = 6, 1 = Channel 6 is the next allocated channel |
| 19 | `NxtChnNo` = 7, 1 = Channel 7 is the next allocated channel |
| 20 | `NxtChnNo` = 8, 1 = Channel 8 is the next allocated channel |
| 21 | `BaEn_5` |
| 22 | `BaEn_6` |
| 23 | `BaEn_7` |
| 24 | `BaEn_8` |
| 25 | `Occ_5` |
| 26 | `Occ_6` |
| 27 | `Occ_7` |
| 28 | `Occ_8` |
| 29 - 31 | Not used |

### See also

ShrdResL modes (Page 1606)
ShrdResL functions (Page 1607)
ShrdResL error handling (Page 1613)
ShrdResL messaging (Page 1613)
ShrdResL block diagram (Page 1619)

## 8.7.2 ShrdResL modes

### ShrdResL operating modes

This block does not have any modes.

### See also

Description of ShrdResL (Page 1603)
ShrdResL functions (Page 1607)
ShrdResL error handling (Page 1613)
ShrdResL messaging (Page 1613)

ShrdResL I/Os (Page 1614)
ShrdResL block diagram (Page 1619)

## 8.7.3 ShrdResL functions

### Functions of ShrdResL

The functions for this block are listed below.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 - 5 | Not used |
| 6 | 1 = Operator can change the priority of channel 1 |
| 7 | 1 = Operator can change the priority of channel 2 |
| 8 | 1 = Operator can change the priority of channel 3 |
| 9 | 1 = Operator can change the priority of channel 4 |
| 10 | 1 = Operator can change the priority of channel 5 |
| 11 | 1 = Operator can change the priority of channel 6 |
| 12 | 1 = Operator can change the priority of channel 7 |
| 13 | 1 = Operator can change the priority of channel 8 |
| 14 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the section Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bit:

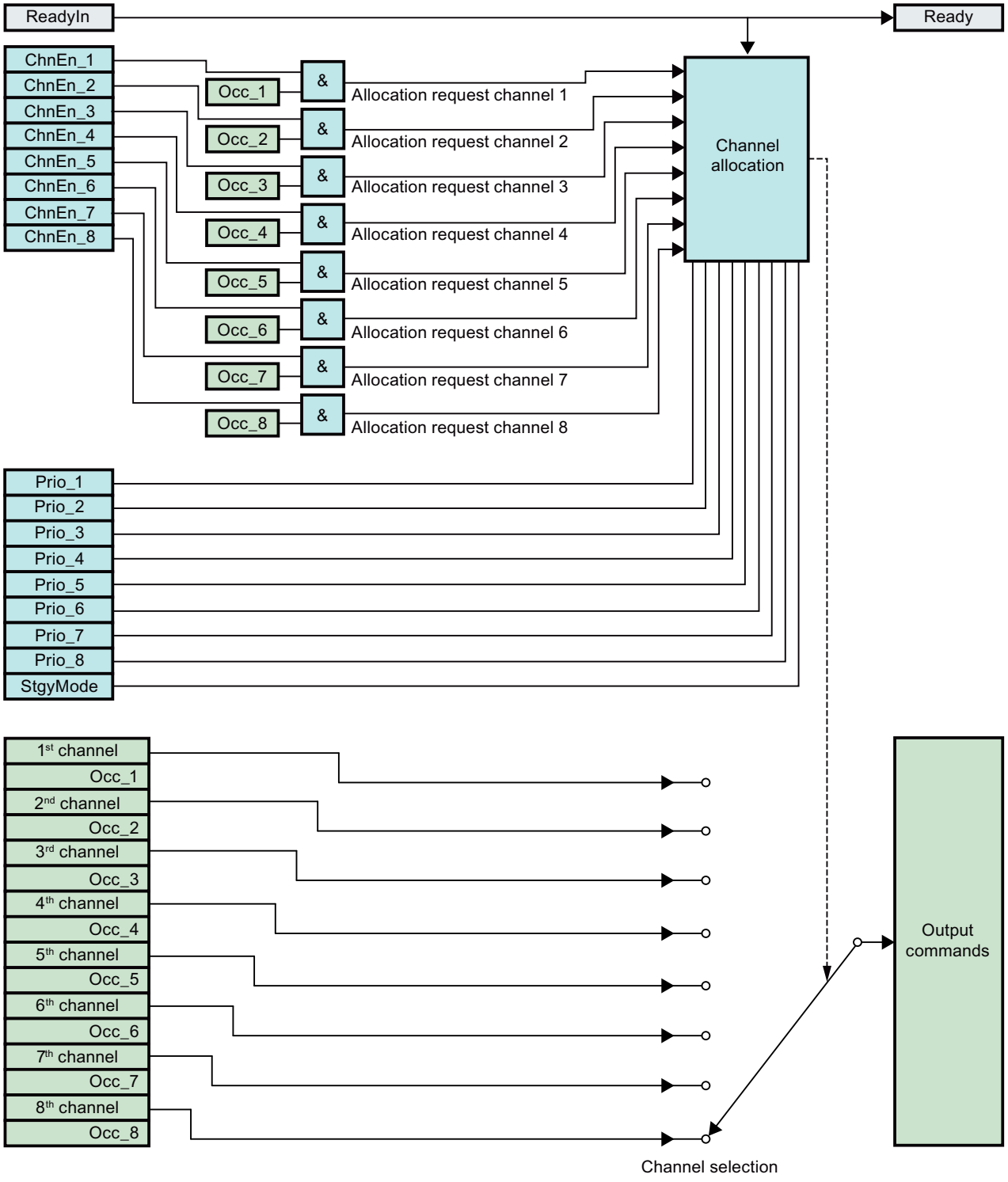
| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |

### Readiness signal

In order to assign one of the channels, the ready signal must be `ReadyIn` = 1 and at least one channel enable must be `ChnEn_x` = 1 (x = 1 to 8). The ready signal is output at the `Ready` output.

## Channel management

Overview of the channel management:



### Allocate/enable channel

A channel can only be allocated if the readiness signal (`ReadyIn` = 1) and the respective channel enabling is available for the allocation `ChnEn_x` = 1.

With `ChnEn_x` = 0, the channel x is disabled and cannot be allocated.

The allocation request of a channel by the upstream application is accomplished via the input `Occ_x`. In case of multiple requests, the channel allocation function stores the requests in a buffer and calculates the allocated channel number with the help of the channel priorities `Prio_x` (x = 1 to 8) and the strategy mode `StgyMode`.

Each channel has its own priority. It can be parameterized at the inputs `Prio_x` (x = 1 to 8).

The strategy mode defines the order of storing the requests in the buffer.

| Strategy 1 | Strategy 2 | Parameter value of `StgyMode` |
|---|---|---|
| First In First Out | Order | 0 |
| First In First Out | Maximum Priority; in case equal request Order is used | 1 |
| First In First Out | Minimum Priority; in case equal request Order is used | 2 |
| Last In First Out | Order | 3 |
| Last In First Out | Maximum Priority; in case equal request Order is used | 4 |
| Last In First Out | Minimum Priority; in case equal request Order is used | 5 |
| Maximum Priority | Order | 6 |
| Maximum Priority | First In First Out | 7 |
| Maximum Priority | Last In First Out | 8 |
| Minimum Priority | Order | 9 |
| Minimum Priority | First In First Out | 10 |
| Minimum Priority | Last In First Out | 11 |

The allocation request will be evaluated with **Strategy 1**. If after the first evaluation, there are equal requests, the **Strategy 2** will be evaluated. This happens in the following cases:

- Two or more requests are in the same cycle and **Strategy 1** is "First In First Out" or "Last In First Out".
- Two or more requests have the same priority and **Strategy 1** is "Maximum Priority" or "Minimum Priority".

The parameterization "Order" in **Strategy 2** means that the priority of the request depends on the channel number:

- Channel 1 has the highest priority
- Channel 8 has the lowest priority

If **Strategy 2** is "Maximum Priority" or "Minimum Priority" and if after the evaluation, there are equal requests, the channel allocation is calculated with "Order".

The applied commands are aligned in accordance with the output command interface.

The number of the allocated channel is displayed at the output `ActChnNo` (INT format). If no channel is allocated, the output is 0.

### Enable/disable channel

If the allocated channel disabling via `Occ_x` = 0 or via `ChnEn_x` = 0, the next channel in the allocation request buffer is automatically switched to the command outputs. If there is no other allocation request (every `Occ_x` = 0), the commands of the output interface are resetted to the default values.

### Channel prioritization

Channel 1 has the highest priority, channel 4 has the lowest. If the input `Occ_x` is set at multiple enabled channels, the channel with the highest priority is allocated and the output `MultiOcc` = 1 is set.

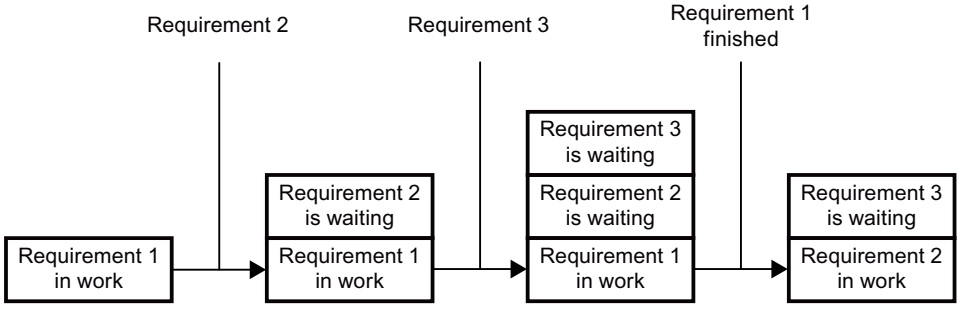
### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Example for "First In First Out" Strategy mode (`StgyMode` = 0, 1, or 2)

The requirement which is issued for the longest duration will be released first.

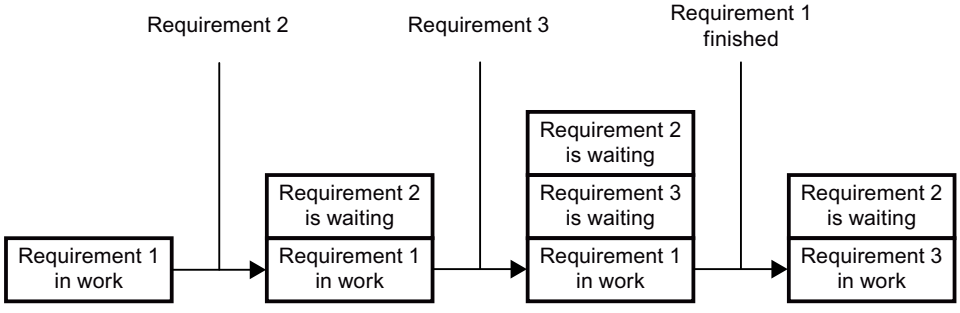
The requirements that are coming in the same cycle will be evaluated after **Strategy 2**.




### Example for "Last In First Out" Strategy mode (`StgyMode` = 3, 4, or 5)

The requirement which is issued for the shortest duration will be released first.

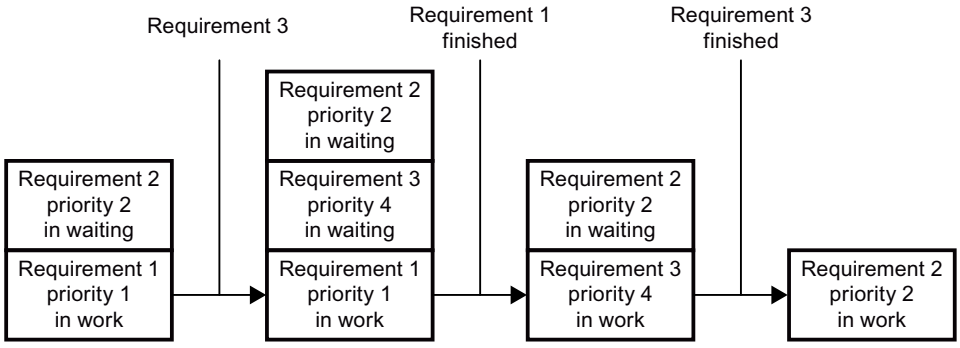
The requirements that are coming in the same cycle will be evaluated after **Strategy 2**.



### Example for "Maximum Priority" Strategy mode (`StgyMode` = 6, 7, or 8)

The requirement with the highest priority will be released first.

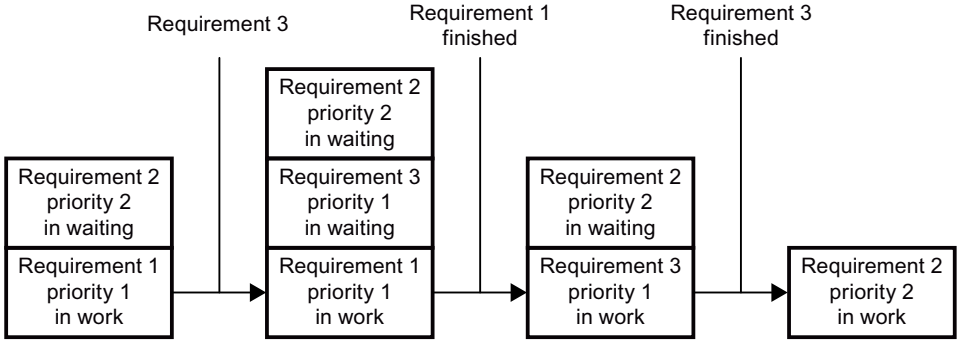
The requirements that are coming in the same cycle will be evaluated after **Strategy 2**.




### Example for "Minimum Priority" Strategy mode (`StgyMode` = 9, 10, or 11)

The requirement with the lowest priority will be released first.

The requirements that are coming in the same cycle will be evaluated after **Strategy 2**.




### See also

Description of ShrdResL (Page 1603)
ShrdResL modes (Page 1606)
ShrdResL error handling (Page 1613)
ShrdResL messaging (Page 1613)
ShrdResL I/Os (Page 1614)
ShrdResL block diagram (Page 1619)

## 8.7.4 ShrdResL error handling

### ShrdResL error handling

Please refer to the section Error handling (Page 128) in the basic instructions.

The following error can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` parameter can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; block is not processed. |
| 0 | There is no error. |
| 49 | Wrong strategy mode number:<br>`StgyMode` < 0 or `StgyMode` > 11 |

### See also

Description of ShrdResL (Page 1603)

ShrdResL modes (Page 1606)

ShrdResL functions (Page 1607)

ShrdResL messaging (Page 1613)

ShrdResL I/Os (Page 1614)

ShrdResL block diagram (Page 1619)

## 8.7.5 ShrdResL messaging

### Messaging

This block does not offer messaging.

### See also

Description of ShrdResL (Page 1603)

ShrdResL modes (Page 1606)

ShrdResL functions (Page 1607)

ShrdResL error handling (Page 1613)

ShrdResL I/Os (Page 1614)
ShrdResL block diagram (Page 1619)

## 8.7.6 ShrdResL I/Os

### I/Os of ShrdResL

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutMod_1* | 1 = "Automatic mode" via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ... | ... | ... | ... |
| AutMod_8* | 1 = "Automatic mode" via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BaEn_1 | 1 = Enable allocation for channel 1 | BOOL | 0 |
| ... | ... | ... | ... |
| BaEn_8 | 1 = Enable allocation for channel 8 | BOOL | 0 |
| BaID_1 | Batch number for channel 1 | DWORD | 16#00000000 |
| ... | ... | ... | ... |
| BaID_8 | Batch number for channel 8 | DWORD | 16#00000000 |
| BaName_1 | Batch designation for channel 1 | S7-String | |
| ... | ... | ... | ... |
| BaName_8 | Batch designation for channel 8 | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| ChnEn_1 | 1 = Channel 1 is enabled and can be allocated<br>0 = Channel 1 is blocked and cannot be allocated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| ... | ... | ... | ... |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ChnEn_8* | 1 = Channel 8 is enabled and can be allocated<br>0 = Channel 8 is blocked and cannot be allocated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| ChnCmd_1 | Channel input 1 (reserved) | STRUCT | |
| ... | ... | ... | ... |
| ChnCmd_8 | Channel input 8 (reserved) | STRUCT | |
| Ctrl01_1* | Control command 1 for channel 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ... | ... | ... | ... |
| Ctrl12_8* | Control command 12 for channel 8 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CtrlW1_1* | Available user control word 1 for channel 1 | DWORD | 16#00000000 |
| ... | ... | ... | ... |
| CtrlW1_8* | Available user control word 1 for channel 8 | DWORD | 16#00000000 |
| CtrlW2_1* | Available user control word 2 for channel 1 | DWORD | 16#00000000 |
| ... | ... | ... | ... |
| CtrlW2_8* | Available user control word 2 for channel 8 | DWORD | 16#00000000 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 1607) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| ManMod_1* | 1 = "Manual mode" via interconnection for channel 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ... | ... | ... | ... |
| ManMod_8* | 1 = "Manual mode" via interconnection for channel 8 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ModLi_1* | 1 = Control via interconnection or SFC for channel 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ... | ... | ... | ... |
| ModLi_8* | 1 = Control via interconnection or SFC for channel 8 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occ_1* | 1 = Occupied<br>It is also used as a control signal for the channel allocation for channel 1 | BOOL | 0 |
| ... | ..0 | ... | ... |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Occ_8* | 1 = Occupied<br>It is also used as a control signal for the channel allocation for channel 8 | BOOL | 0 |
| Prio_1 | Priority for channel 1 | INT | 1 |
| ... | ... | ... | ... |
| Prio_8 | Priority for channel 8 | INT | 8 |
| ReadyIn | Readiness signal:<br>1 = Signal for channel enabling active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstLi_1* | 1 = Resetting via interconnection for channel 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ... | ... | ... | ... |
| RstLi_8* | 1 = Resetting via interconnection for channel 8 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SelFpRes | Selection for faceplate resource | ANY | |
| SP_Ex_1 | 1 = Select external setpoint (via interconnection) for channel 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ... | ... | ... | ... |
| SP_Ex_8 | 1 = Select external setpoint (via interconnection) for channel 8 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_In_1* | 1 = Select internal setpoint (via interconnection) for channel 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ... | ... | ... | ... |
| SP_In_8* | 1 = Select internal setpoint (via interconnection) for channel 8 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP1Ext_1* | External setpoint 1 for channel 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ... | ... | ... | ... |
| SP8Ext_8* | External setpoint 8 for channel 8 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| StepNo_1 | Step number for channel 1 | DWORD | 16#00000000 |
| ... | ... | ... | ... |
| StepNo_8 | Step number for channel 8 | DWORD | 16#00000000 |
| StgyMode | Strategy mode (0 to 11) | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Strng1_1 | Additional string 1 for channel 1 | S7-String | |
| Strng2_1 | Additional string 2 for channel 1 | S7-String | |
| ... | ... | ... | ... |
| Strng1_8 | Additional string 1 for channel 8 | S7-String | |
| Strng2_8 | Additional string 2 for channel 8 | S7-String | |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 1607) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| UserStatus | Freely assignable bits for use in PCS 7 OS | BYTE | 16#00 |
| UserStatus2 | Freely assignable bits for use in PCS 7 OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ActChnNo | Displays the allocated channel | INT | 0 |
| AutMod | 1= "Automatic mode" via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| Ctrl01 | Control command 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ... | ... | ... | ... |
| Ctrl12 | Control command 12 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CtrlW1 | Available user control word 1 | DWORD | 16#00000000 |
| CtrlW2 | Available user control word 2 | DWORD | 16#00000000 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of the current error number. For error numbers that can be outputted by this block, see ShrdResL error handling (Page 1613) | INT | -1 |
| NxtChnNo | Next occupied channel number | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ManMod | 1 = "Manual mode" via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ModLi | 1 = Control via interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MultiOcc | 1 = More than one channel request is present | BOOL | 0 |
| Occupied | 1 = Occupied<br>Is also used as a control signal for the channel allocation. | BOOL | 0 |
| Ready | 1 = Active readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstLi | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtLi | 1 = Select external setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_IntLi | 1 = Select internal setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP1Ext | External setpoint 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ... | ... | ... | ... |
| SP8Ext | External setpoint 8 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Status1 | Status word 1 (Page 1603) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1603) | DWORD | 16#00000000 |
| StepNo | Step number | DWORD | 16#00000000 |
| Strng1 | Additional string 1 | S7-String | |
| Strng2 | Additional string 2 | S7-String | |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |

**See also**

ShrdResL modes (Page 1606)

ShrdResL messaging (Page 1613)

ShrdResL block diagram (Page 1619)

## 8.7.7 ShrdResL block diagram

### ShrdResL block diagram

A block diagram is not provided for this block.

**See also**

Description of ShrdResL (Page 1603)

ShrdResL modes (Page 1606)

ShrdResL functions (Page 1607)

ShrdResL error handling (Page 1613)

ShrdResL messaging (Page 1613)

ShrdResL I/Os (Page 1614)

## 8.7.8 Operator control and monitoring

### 8.7.8.1 ShrdResL views

### Views of the ShrdResL block

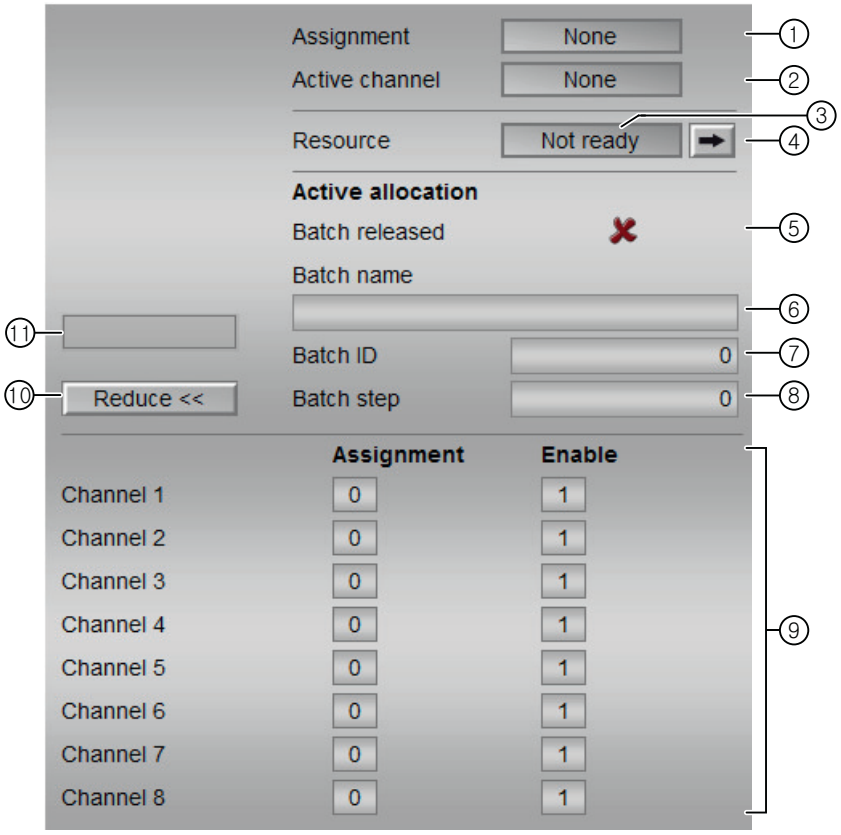
The block ShrdResL provides the following views:

- ShrdResL standard view (Page 1620)
- ShrdResL general preview (Page 1622)
- ShrdResL preview (Page 1622)
- ShrdResL parameter view (Page 1624)
- Memo view (Page 338)
- Block icon for ShrdResL (Page 1625)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 8.7.8.2 ShrdResL standard view

#### ShrdResL standard view




#### (1) Allocation

The allocation status is displayed in this area:

- "None": Display for all `Occ_1`...`Occ_4` = 0 and `Occupied` = 0
- "Requested": Display for one of `Occ_1`...`Occ_4` = 1 and `Occupied` = 0
- "Active": Display for `Occupied` = 1

#### (2) Active channel

The channel number of the active channel is displayed in this area.

If the text is configured for this command (Text 1 in the object properties), it is displayed as additional text and button label for command selection. Additional information is available in the section Labeling of buttons and text (Page 235).

### (3) Resource

The status of the enable signal is displayed in this area:

- "Idle": Display for `ReadyIn` = 1
- "Not ready": Display for `ReadyIn` = 0

### (4) Jump key to standard view of any faceplate

This display is only visible when the corresponding block input is interconnected.

Use this navigation button to jump to the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find more information on this in the section Opening additional faceplates (Page 234).

### (5) Release batch

This area displays if the block is released for operation via SIMATIC BATCH (`BatchEn` = 1).

### (6) Batch name

This area displays the name of the batch that is currently running (`Batchname`).

### (7) Batch ID

This area displays the identification number of the batch that is currently running (`BatchID`).

### (8) Batch step

This area displays the step number of the batch that is currently running (`StepNo`).

### (9) Display channel 1 to 8

This area is only visible when the **(10)** "Expand" button is pressed.

The "Allocation" and "Release" status of the channels 1 to 8 are displayed in this area.

The next channel for allocation, which is in waiting position, is marked with a yellow exclamation mark.

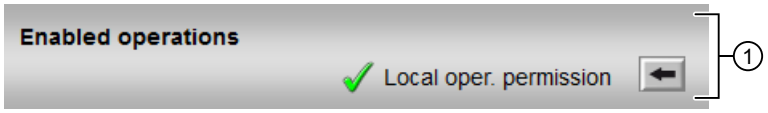
### (11) Display area for block states

This area provides additional information on the operating state of the block:

- "Internal error"
- "Invalid Signal"

### 8.7.8.3 ShrdResL general preview

#### ShrdResL general preview




#### (1) Enabled operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

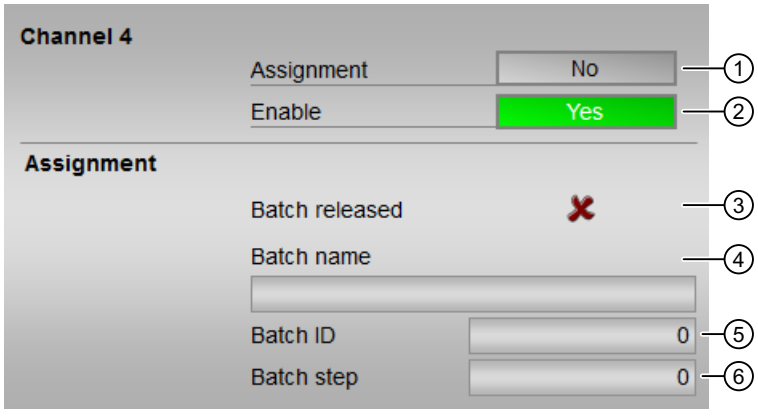
- **Green check mark**: The OS operator can control this parameter.
- **Gray check mark**: The OS operator cannot control this parameter at this time due to the process.
- **Red cross**: The OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm`).

The following enabled operation is shown here:

- "Local operator permission": Use the ← button to switch to the standard view of the OpStations block. You can find additional information on this in the section Operator control permissions (Page 285).

### 8.7.8.4 ShrdResL preview

#### ShrdResL preview




Each of the eight channels has its own preview. The previews of the individual channels are identical. The preview shown in this graphic is of channel 4.

### (1) Allocation

The allocation status of the channel is displayed in this area.

- "No": Display for `Occ_1` = 0
- "Requested": Display for `Occ_1` = 1and `ActChnNo` = 0
- "Active": Display for `ActChnNo` = 1

### (2) Enable

The status of the enable is displayed in this area.

- "Yes": Display for `ChnEn` = 1
- "No": Display for `ChnEn` = 0

### (3) Release batch

This area displays if the block is released for operation via SIMATIC BATCH (`BatchEn` = 1).

### (4) Batch name

This area displays the name of the batch that is currently running (`Batchname`).

### (5) Batch ID

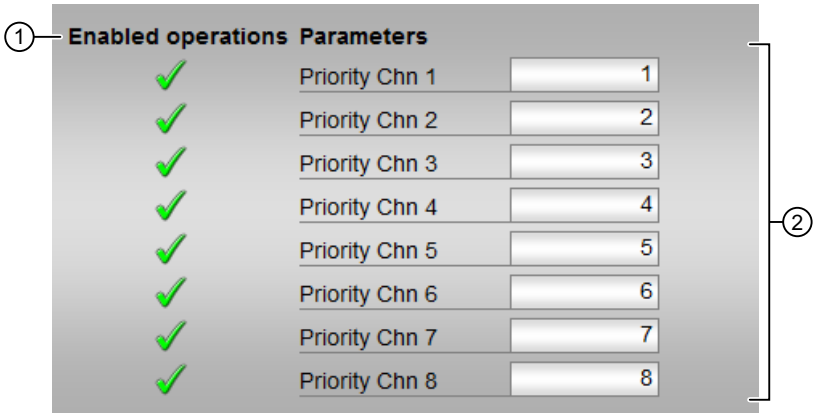
This area displays the identification number of the batch that is currently running (`BatchID`).

### (6) Batch step

This area displays the step number of the batch that is currently running (`StepNo`).

### 8.7.8.5 ShrdResL parameter view

#### Parameter view of ShrdResL




#### (1) Enabled operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for operation enable:

- **Green check mark**: The OS operator can control this parameter.
- **Gray check mark**: The OS operator cannot control this parameter at this time due to the process.
- **Red cross**: The OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm`).

#### (2) Parameters

In this area, you can change the parameters and thereby influence the behavior of ShrdResL. You can find more information on this in the section Changing values (Page 291).

You can influence the following parameters:

- "Priority Chn 1": Priority for channel 1
- "Priority Chn 2": Priority for channel 2
- "Priority Chn 3": Priority for channel 3
- "Priority Chn 4": Priority for channel 4
- "Priority Chn 5": Priority for channel 5
- "Priority Chn 6": Priority for channel 6
- "Priority Chn 7": Priority for channel 7
- "Priority Chn 8": Priority for channel 8

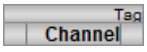
### 8.7.8.6 Block icon for ShrdResL

## Block icons for ShrdResL

A variety of block icons are available with the following functions:

- Display active channel
- Process tag type (2 only)
- Memo display (2 only)
- Fixed text (language dependent, 2 only)

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

# 8.8 ShrdResS - Multiplexer for shared resources (Small)

## 8.8.1 Description for ShrdResS

### Object name (type + number) and family

Type + number: FB 1914

Family: Drives

### Area of application for ShrdResS

The block is used for the following applications:

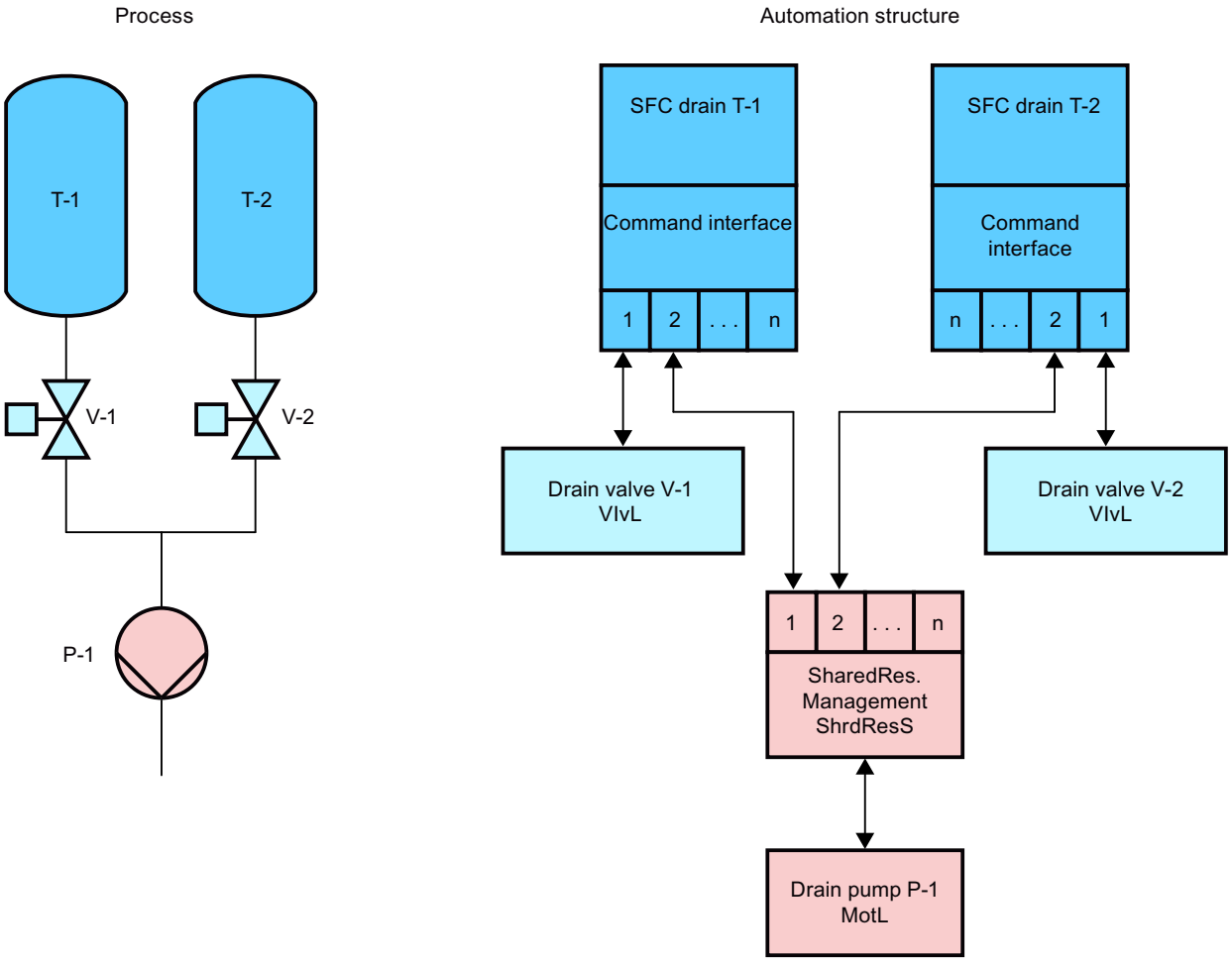
- Organize individual requests for downstream block from upstream applications

### How it works

The block coordinates the access from up to four SFC step sequencers to a technologic block of the families "Drives" or "Dose". These blocks include:

- Motors
- Valves
- Dosers

The block has four channels, each with a standardized command interface.




The block can be cascaded via the fourth channel so that accesses via more than four upstream applications are possible. Use the cascade by interconnecting the output parameter `CasOut` of the first block to the input parameter `CasIn` of the second block. The output interface of the upstream block is then used as channel 4 for the second block.

As soon as a channel is allocated, its command interface is aligned 1-to-1 with the command interface at the output.

The channel with the lower number always has priority with shared allocation. Cascaded blocks are always allocated to channel 4 of the upstream block; they are then in 4th place.

## Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

## Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

### Status word allocation for `Status` parameter

You can find a description for each parameter in section ShrdResS I/Os (Page 1633).

| Status bit | Parameter |
|---|---|
| 0 | `ReadyIn`, readiness signal of the interconnected technologic block |
| 1 | `ChnEn_1` |
| 2 | `ChnEn_2` |
| 3 | `ChnEn_3` |
| 4 | `ChnEn_4` |
| 5 | `Ready`, channel allocation is possible |
| 6 | `ActChnNo` = 1, 1 = Channel 1 enabled |
| 7 | `ActChnNo` = 2, 1 = Channel 2 enabled |
| 8 | `ActChnNo` = 3, 1 = Channel 3 enabled |
| 9 | `ActChnNo` = 4, 1 = Channel 4 enabled |
| 10 | `MultiOcc`, 1 = Request for more than one channel |
| 11 | `Cascaded`, 1 = 4. Channel is cascaded |
| 12 | `BatchEn` |
| 13 | `Occupied` |
| 14 - 20 | Not used |
| 21 | `BaEn_1` |
| 22 | `BaEn_2` |
| 23 | `BaEn_3` |
| 24 | `BaEn_4` |
| 25 | `Occ_1` |
| 26 | `Occ_2` |
| 27 | `Occ_3` |
| 28 | `Occ_4` |
| 29 - 31 | Not used |

### Status word allocation for `Status1` parameter

| Status bit | Parameter |
|---|---|
| 31 | Internal error |

### See also

ShrdResS modes (Page 1629)

ShrdResS functions (Page 1629)

ShrdResS error handling (Page 1632)

ShrdResS messaging (Page 1632)

ShrdResS block diagram (Page 1641)

## 8.8.2 ShrdResS modes

### ShrdResS operating modes

This block does not have any modes.

### See also

Description for ShrdResS (Page 1626)

ShrdResS functions (Page 1629)

ShrdResS error handling (Page 1632)

ShrdResS messaging (Page 1632)

ShrdResS I/Os (Page 1633)

ShrdResS block diagram (Page 1641)

## 8.8.3 ShrdResS functions

### Functions of ShrdResS

The functions for this block are listed below.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

If the block is cascaded, the standard view of the ShrdResS block interconnected to `CasIn` can be opened in addition to the freely configured faceplate.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the section Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

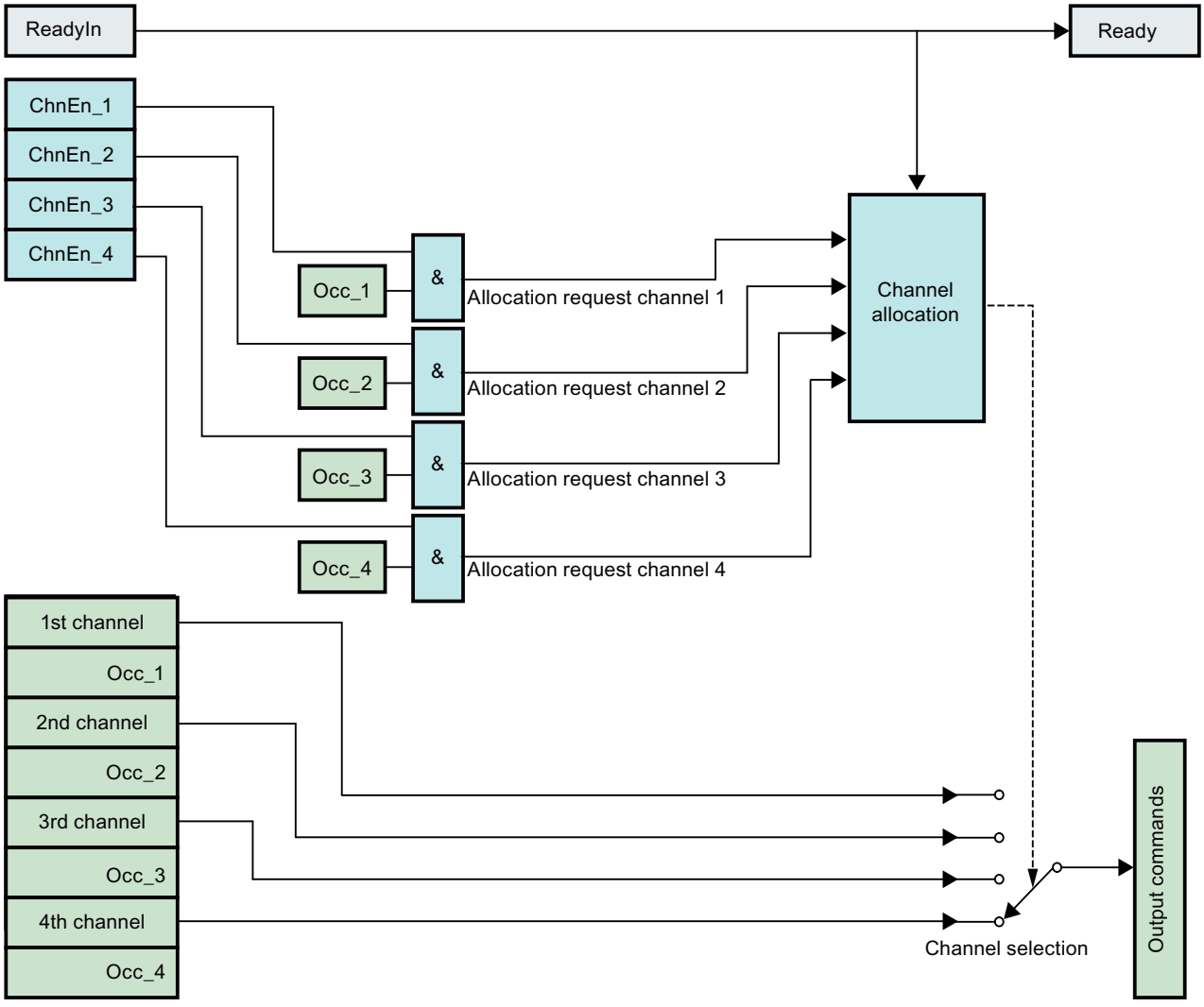
| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |

### Readiness signal

In order to assign one of the channels, the ready signal must be `ReadyIn` = 1 and at least one channel enable must be `ChnEn_x = 1` (x = 1...4) . The ready signal is output at the `Ready` output.

## Channel management

Overview of the channel management




## Allocate/enable channel

A channel can only be allocated if the readiness signal (`ReadyIn` = 1) and the respective channel enabling is available for the allocation `ChnEn_x` = 1.

With `ChnEn_x` = 0 the channel x is disabled and cannot be allocated.

The allocation of a channel by the upstream application is accomplished via the input `Occ_x`. As long as this input is 1, the channel is allocated and enabled. The applied commands are aligned 1-to-1 with the output command interface.

The number of the allocated channel is displayed at the output `ActChnNo` (INT format). If no channel is allocated, the output is 0.

### Enable/disable channel

If no channel enabling via `Occ_x` = 0 occurs or if the channel is disabled via `ChnEn_x` = 0, the next highest priority channel is automatically switched to the command outputs. If there is no other allocation request (every `Occ_x` = 0), the commands of the output interface are reset to the default values.

### Channel prioritization

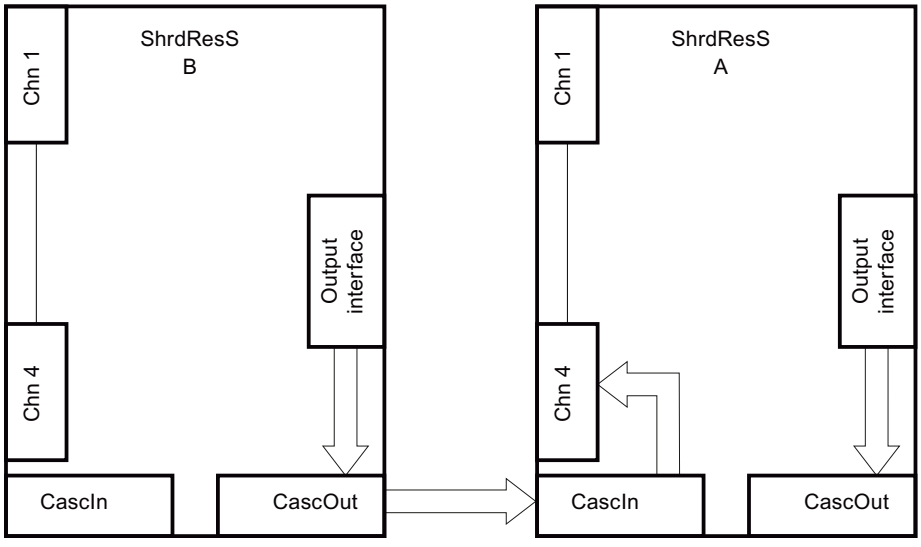
Channel 1 has the highest priority, channel 4 has the lowest. If the input `Occ_x` is set at multiple enabled channels, the channel with the highest priority is allocated and the output `MultiOcc` = 1 is set.

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Cascading

For cascading, the output `CasOut` of a ShrdResS block B must be connected with the input `CasIn` of the downstream ShrdResS block A. Therefore, the fourth channel of the downstream block ShrdResS A is allocated by the ShrdResS B block connected to `CasIn`. The command interface of the fourth channel is tracked in this case to the output interface of the connected ShrdResS B block.




Any values applied via interconnection at the 4th channel of block A are not taken into account in the block code during cascading.

### See also

Description for ShrdResS (Page 1626)

ShrdResS modes (Page 1629)

ShrdResS error handling (Page 1632)
ShrdResS messaging (Page 1632)
ShrdResS I/Os (Page 1633)
ShrdResS block diagram (Page 1641)

## 8.8.4 ShrdResS error handling

### ShrdResS error handling

Please refer to the section Error handling (Page 128) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; block will not be processed |
| 0 | There is no error. |
| 40 | Interconnection error on `CasIn`, incorrect block type interconnected |

### See also

Description for ShrdResS (Page 1626)
ShrdResS modes (Page 1629)
ShrdResS functions (Page 1629)
ShrdResS messaging (Page 1632)
ShrdResS I/Os (Page 1633)
ShrdResS block diagram (Page 1641)

## 8.8.5 ShrdResS messaging

### Messaging

This block does not offer messaging.

### See also

Description for ShrdResS (Page 1626)
ShrdResS modes (Page 1629)
ShrdResS functions (Page 1629)
ShrdResS error handling (Page 1632)
ShrdResS I/Os (Page 1633)
ShrdResS block diagram (Page 1641)

## 8.8.6 ShrdResS I/Os

### I/Os of ShrdResS

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutMod_1* | 1= "Automatic mode" via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutMod_2* | 1= "Automatic mode" via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutMod_3* | 1= "Automatic mode" via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutMod_4* | 1= "Automatic mode" via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BaEn_1 | 1 = Enable allocation for channel 1 | BOOL | 0 |
| BaEn_2 | 1 = Enable allocation for channel 2 | BOOL | 0 |
| BaEn_3 | 1 = Enable allocation for channel 3 | BOOL | 0 |
| BaEn_4 | 1 = Enable allocation for channel 4 | BOOL | 0 |
| BaID_1 | Batch number for channel 1 | DWORD | 16#00000000 |
| BaID_2 | Batch number for channel 2 | DWORD | 16#00000000 |
| BaID_3 | Batch number for channel 3 | DWORD | 16#00000000 |
| BaID_4 | Batch number for channel 4 | DWORD | 16#00000000 |
| BaName_1 | Batch designation for channel 1 | S7-String | |
| BaName_2 | Batch designation for channel 2 | S7-String | |
| BaName_3 | Batch designation for channel 3 | S7-String | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| BaName_4 | Batch designation for channel 4 | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| CasIn | Input for cascade, to be interconnected with the output parameter CasOut of the preceding ShrdResS block | ANY | |
| ChnEn_1 | 1 = Channel 1 is enabled and can be allocated<br>0 = Channel 1 is blocked and cannot be allocated. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| ChnEn_2 | 1 = Channel 2 is enabled and can be allocated<br>0 = Channel 2 is blocked and cannot be allocated. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| ChnEn_3 | 1 = Channel 3 is enabled and can be allocated<br>0 = Channel 3 is blocked and cannot be allocated. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| ChnEn_4* | 1 = Channel 4 is enabled and can be allocated<br>0 = Channel 4 is blocked and cannot be allocated. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| ChnCmd_1 | Channel 1: Command interface (reserved) | STRUCT | |
| ChnCmd_2 | Channel 2: Command interface (reserved) | STRUCT | |
| ChnCmd_3 | Channel 3: Command interface (reserved) | STRUCT | |
| ChnCmd_4 | Channel 4: Command interface (reserved) | STRUCT | |
| Ctrl01_1* | 1. Control command for channel 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl02_1* | 2. Control command for channel 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Ctrl03_1* | 3. Control command for channel 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl04_1* | 4. Control command for channel 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl05_1* | 5. Control command for channel 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl06_1* | 6. Control command for channel 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl01_2* | 1. Control command for channel 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl02_2* | 2. Control command for channel 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl03_2* | 3. Control command for channel 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl04_2* | 4. Control command for channel 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl05_2* | 5. Control command for channel 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl06_2* | 6. Control command for channel 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl01_3* | 1. Control command for channel 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl02_3* | 2. Control command for channel 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl03_3* | 3. Control command for channel 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Ctrl04_3* | 4. Control command for channel 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl05_3* | 5. Control command for channel 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl06_3* | 6. Control command for channel 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl01_4* | 1. Control command for channel 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl02_4* | 2. Control command for channel 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl03_4* | 3. Control command for channel 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl04_4* | 4. Control command for channel 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl05_4* | 5. Control command for channel 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl06_4* | 6. Control command for channel 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CtrlW1_1* | 1. Available user control word for channel 1 | DWORD | 16#00000000 |
| CtrlW1_2* | 1. Available user control word for channel 2 | DWORD | 16#00000000 |
| CtrlW1_3* | 1. Available user control word for channel 3 | DWORD | 16#00000000 |
| CtrlW1_4* | 1. Available user control word for channel 4 | DWORD | 16#00000000 |
| CtrlW2_1* | 2. Available user control word for channel 1 | DWORD | 16#00000000 |
| CtrlW2_2* | 2. Available user control word for channel 2 | DWORD | 16#00000000 |
| CtrlW2_3* | 2. Available user control word for channel 3 | DWORD | 16#00000000 |
| CtrlW2_4* | 2. Available user control word for channel 4 | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 1629) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| ManMod_1* | 1 = "Manual mode" via interconnection for channel 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManMod_2* | 1 = "Manual mode" via interconnection for channel 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManMod_3* | 1 = "Manual mode" via interconnection for channel 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManMod_4* | 1 = "Manual mode" via interconnection for channel 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ModLi_1* | 1 = Control via interconnection or SFC for channel 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ModLi_2* | 1 = Control via interconnection or SFC for channel 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ModLi_3* | 1 = Control via interconnection or SFC for channel 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ModLi_4* | 1 = Control via interconnection or SFC for channel 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occ_1 | 1 = Occupied<br>Is also used as a control signal for the channel allocation for channel 1 | BOOL | 0 |
| Occ_2 | 1 = Occupied<br>Is also used as a control signal for the channel allocation for channel 2 | BOOL | 0 |
| Occ_3 | 1 = Occupied<br>Is also used as a control signal for the channel allocation for channel 3 | BOOL | 0 |
| Occ_4* | 1 = Occupied<br>Is also used as a control signal for the channel allocation for channel 4 | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ReadyIn | Readiness signal:<br>1 = Signal for channel enabling active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstLi_1* | 1 = Resetting via interconnection for channel 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstLi_2* | 1 = Resetting via interconnection for channel 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstLi_3* | 1 = Resetting via interconnection for channel 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstLi_4* | 1 = Resetting via interconnection for channel 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SelFpRes | Selection for faceplate resource | ANY | |
| SP_Ex_1 | 1 = Select external setpoint (via interconnection) for channel 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_Ex_2 | 1 = Select external setpoint (via interconnection) for channel 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_Ex_3 | 1 = Select external setpoint (via interconnection) for channel 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_Ex_4 | 1 = Select external setpoint (via interconnection) for channel 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_In_1* | 1 = Select internal setpoint (via interconnection) for channel 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_In_2* | 1 = Select internal setpoint (via interconnection) for channel 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_In_3* | 1 = Select internal setpoint (via interconnection) for channel 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_In_4* | 1 = Select internal setpoint (via interconnection) for channel 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP1Ext_1* | 1. External setpoint for channel 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP2Ext_1* | 2. External setpoint for channel 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP1Ext_2* | 1. External setpoint for channel 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP2Ext_2* | 2. External setpoint for channel 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP1Ext_3* | 1. External setpoint for channel 3 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP2Ext_3* | 2. External setpoint for channel 3 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP1Ext_4* | 1. External setpoint for channel 4 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP2Ext_4* | 2. External setpoint for channel 4 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| StepNo_1 | Step number for channel 1 | DWORD | 16#00000000 |
| StepNo_2 | Step number for channel 2 | DWORD | 16#00000000 |
| StepNo_3 | Step number for channel 3 | DWORD | 16#00000000 |
| StepNo_4 | Step number for channel 4 | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ActChnNo | Displaying the allocated channel | INT | 0 |
| AutMod | 1= "Automatic mode" via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Cascaded | 1 = Block is cascaded | BOOL | 0 |
| CasOut | 1 = Output parameter for cascade formation, to be interconnected with the input parameter CasIn of the following ShrdResS block | DWORD | 16#00000000 |
| Ctrl01 | 1. Control command | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl02 | 2. Control command | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl03 | 3. Control command | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl04 | 4. Control command | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl05 | 5. Control command | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl06 | 6. Control command | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CtrlW1 | 1. Available user control word | DWORD | 16#00000000 |
| CtrlW2 | 2. Available user control word | DWORD | 16#00000000 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see ShrdResS error handling (Page 1632) | INT | -1 |
| ManMod | 1 = "Manual mode" via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ModLi | 1 = Control via interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MultiOcc | 1 = More than one channel request is present | BOOL | 0 |
| Occupied | 1 = Occupied<br>Is also used as a control signal for the channel allocation | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Ready | 1 = Active readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstLi | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ExtLi | 1 = Select external setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_IntLi | 1 = Select internal setpoint (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP1Ext | 1. External setpoint | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP2Ext | 2. External setpoint | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Status1 | Status word 1 (Page 1626) | DWORD | 16#00000000 |
| StepNo | Step number | DWORD | 16#00000000 |

### See also

ShrdResS modes (Page 1629)

ShrdResS messaging (Page 1632)

ShrdResS block diagram (Page 1641)

## 8.8.7 ShrdResS block diagram

### ShrdResS block diagram

A block diagram is not provided for this block.

### See also

Description for ShrdResS (Page 1626)

ShrdResS modes (Page 1629)

ShrdResS functions (Page 1629)

ShrdResS error handling (Page 1632)

ShrdResS messaging (Page 1632)

ShrdResS I/Os (Page 1633)

## 8.8.8 Operator control and monitoring

### 8.8.8.1 ShrdResS views

#### Views of the ShrdResS block

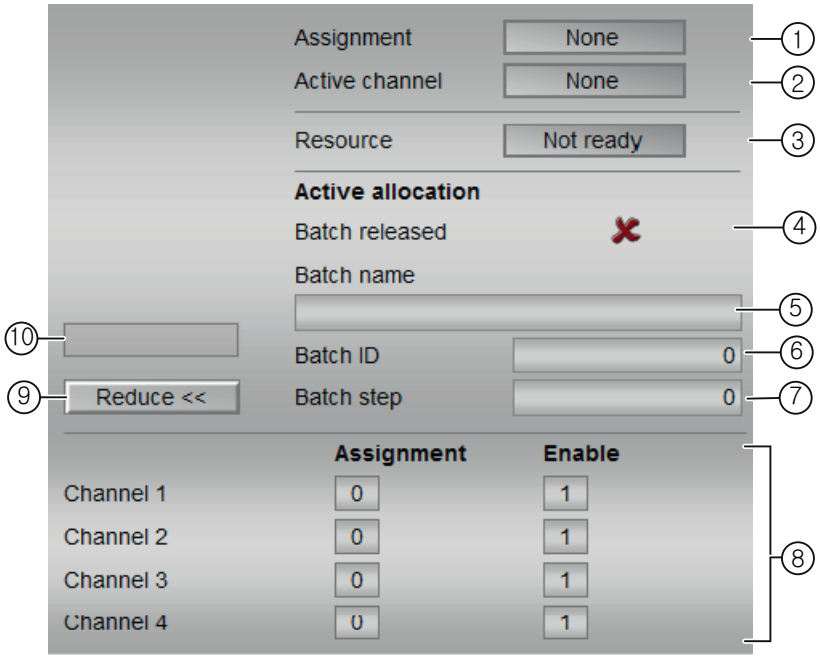
The block ShrdResS provides the following views:

- ShrdResS standard view (Page 1642)
- ShrdResS preview (Page 1644)
- Memo view (Page 338)
- Block icon for ShrdResS (Page 1645)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 8.8.8.2 ShrdResS standard view

#### ShrdResS standard view



### (1) Allocation

The allocation status is displayed in this area.

- "None": Display for `Status1.Bit 10 = 0` and `Status1.Bit 13 = 0`
- "Requested": Display for `Status1.Bit 10 = 1` and `Status1.Bit 13 = 0`
- "Active": Display for `Status1.Bit 13 = 1`

### (2) Active channel

The channel number of the active channel is displayed in this area.

If text is configured for this command (Text 1 in the object properties), it is displayed as additional text and button label for command selection. Additional information is available in the section Labeling of buttons and text (Page 235).

Use the ← button to switch to the standard view of the connected `CasIn` block.

Use the → button to switch to the standard view of the connected `CasOut` block.

### (3) Resource

The status of the general enable signal is displayed in this area.

- "Idle": Display for `ReadyIn = 1`
- "Not ready": Display for `ReadyIn = 0`

Use the → button to switch to the standard view of the connected `SelFpRes` block.

### (4) Release batch:

This area shows you if the block is released for operation via SIMATIC BATCH (`BatchEn = 1`).

### (5) Batch name

This area shows the name of the batch that is currently running (`Batchname`).

### (6) Batch ID

This area shows the identification number of the batch that is currently running (`BatchID`).

### (7) Batch step

This area shows you the step number of the batch that is currently running (`StepNo`).

### (8) Display channel 1-4

This area is only visible when the **(9)** "Expand" button is pressed.

The "Allocation" und "Release" status of channels 1-4 are displayed in this area.

### (9) Expand / Collapse

This button enables or disables the display area **(8)**. The label of the button changes accordingly.

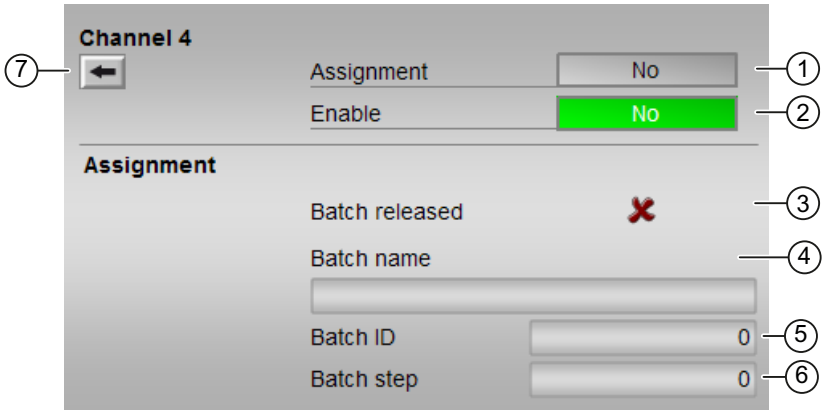
### (10) Display area for block states

This area provides additional information on the operating state of the block:

- "Internal error"
- "Invalid Signal"

## 8.8.8.3 ShrdResS preview

### ShrdResS preview




Each of the four channels has its own preview. The previews of the individual channels are identical except for number **(7)** . The preview described here is based on channel 4.

### (1) Allocation

The allocation status of the channel is displayed in this area.

- "No": Display for `Occ_1 = 0`
- "Requested": Display for `Occ_1 = 1` and `ActChnNo = 0`
- "Active": Display for `ActChnNo = 1`

### (2) Enable

The status of the enable is displayed in this area.

- "Yes": Display for `ChnEn = 1`
- "No": Display for `ChnEn = 0`

**(3) Release batch:**

This area shows you if the block is released for operation via SIMATIC BATCH (`BatchEn = 1`).

**(4) Batch name**

This area shows the name of the batch that is currently running (`Batchname`).

**(5) Batch ID**

This area shows the identification number of the batch that is currently running (`BatchID`).

**(6) Batch step**

This area shows you the step number of the batch that is currently running (`StepNo`).

**(7) Button ←**

Use the ← button to switch to the standard view of the cascaded ShrdResS block. This button is only available for channel 4.

### 8.8.8.4 Block icon for ShrdResS

**Block icons for ShrdResS**

A variety of block icons are available with the following functions:

- Display active channel
- Process tag type (2 only)
- Memo display (2 only)
- Fixed text (language dependent, 2 only)

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

# 8.9 Vlv2WayL - Two-way valve

## 8.9.1 Description of Vlv2WayL

### Object name (type + number) and family

Type + number: FB 1897

Family: Drives

### Area of application for Vlv2WayL

The block is used for the following applications:

- Control of multi-way valves with up to three switching positions. One of these positions is the neutral position (de-energized position)
- Control of three individual valves (valve network) to implement a 2-way valve circuit with neutral position (de-energized position)

### How it works

The multi-way valve (or valve network) is controlled via position 0 (neutral position), position 1 (way 1), or position 2 (way 2). Various inputs are available for controlling the positions.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

For the Vlv2WayL block, the Advanced Process Library contains a template for process tag types as an example with an application scenario for this block.

Example of process tag types:

- Two-way valve (Valve2Way) (Page 2866)

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section Vlv2WayL I/Os (Page 1664).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `AutoAct.Value` |
| 6 | `LocalAct.Value` |
| 7 | 0: Open padlock in the block icon<br>1: Closed padlock in the block icon |
| 8 | "Open"/"Closed" command for `V0` |
| 9 | "Open"/"Closed" command for `V1` |
| 10 | "Open"/"Closed" command for `V2` |
| 11 | Feedback error without control change |
| 12 | Feedback error due to control change |
| 13 | `BypProt` active |
| 14 | Invalid signal status |
| 15 | Mode switchover error |
| 16 | 1 = `Intlock` is active |
| 17 | 1 = `Permit` is active |
| 18 | 1 = `Protect` is active |
| 19 | Display „Force Pos0“ |
| 20 | Display „Force Pos1“ |
| 21 | Display „Force Pos2“ |
| 22 | Feedback for Pos0 OK |
| 23 | Feedback for Pos1 OK |
| 24 | Feedback for Pos2 OK |
| 25 | Feedback for current position OK |
| 26 | Automatic preview Pos0 |
| 27 | Automatic preview Pos1 |
| 28 | Automatic preview Pos2 |
| 29 | `SafeV0` |
| 30 | `SafeV1` |
| 31 | `SafeV2` |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `MsgLock` |
| 1 | Forcing active |

| Status bit | Parameter |
|---|---|
| 2 | Display for interlocks in block icon |
| 3 | `WarnAct.Value` |
| 4 | External error generated by FaultExt or external control system fault from CSF with set `Feature.Bit18` Activating error state for external process control error CSF (Page 162) |
| 5 | For the status display error in the valve |
| 6 | Current monitoring time `P0` is visible |
| 7 | Current monitoring time `V0` is visible |
| 8 | Current monitoring time `V1` is visible |
| 9 | Current monitoring time `V2` is visible |
| 10 | Current monitoring time `P1` is visible |
| 11 | Current monitoring time `P2` is visible |
| 12 | 1 = Input parameter `FbkP1` is connected and `Feature.Bit12` = 1 |
| 13 | 1 = Input parameter `FbkP2` is connected and `Feature.Bit12` = 1 |
| 14 | 1 = Input parameter `FbkP0` is connected |
| 15 | 1 = Input parameter `FbkV0` is connected and `Feature.Bit12` = 0 |
| 16 | 1 = Input parameter `FbkV1` is connected and `Feature.Bit12` = 0 |
| 17 | 1 = Input parameter `FbkV2` is connected and `Feature.Bit18` = 0 |
| 18 | Reset request in automatic |
| 19 | 1 = No impact of input signals on "local mode" with `LocalSetting` = 0, `LocalSetting` = 2, `LocalSetting` = 4 and OOS mode. |
| 20 | `CtrlV0.Value` |
| 21 | `CtrlV1.Value` |
| 22 | `CtrlV2.Value` |
| 23 | `FbkV0Out.Value` |
| 24 | `FbkV1Out.Value` |
| 25 | `FbkV2Out.Value` |
| 26 | `FbkP0Out.Value` |
| 27 | Feedback `V0` (`FbkV0`), for OS display only |
| 28 | Feedback `V1` (`FbkV1`), for OS display only |
| 29 | Feedback `V2` (`FbkV2`), for OS display only |
| 30 | Bypass information for display in block icon |
| 31 | `MS_RelOp` |

## Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 | "Interlock" button is enabled |
| 1 | "Permission" button is enabled |
| 2 | "Protection" button is enabled |
| 3 | `Pos0Out` |
| 4 | Travel in position 0 |
| 5 | Monitoring error in position 0 |

| Status bit | Parameter |
|---|---|
| 6 | `Pos1Out` |
| 7 | Travel in position 1 |
| 8 | Monitoring error in position 1 |
| 9 | `Pos2Out` |
| 10 | Travel in position 2 |
| 11 | Monitoring error in position 2 |
| 12 | Preview position 0 control `CtrlV0` |
| 13 | Preview position 0 control `CtrlV1` |
| 14 | Preview position 0 control `CtrlV2` |
| 15 | Preview position 1 control `CtrlV0` |
| 16 | Preview position 1 control `CtrlV1` |
| 17 | Preview position 1 control `CtrlV2` |
| 18 | Preview position 2 control `CtrlV0` |
| 19 | Preview position 2 control `CtrlV1` |
| 20 | Preview position 2 control `CtrlV2` |
| 21 | Preview for automatic control `CtrlV0` |
| 22 | Preview for automatic control `CtrlV1` |
| 23 | Preview for automatic control `CtrlV2` |
| 24 | `UserAna1` interconnected |
| 25 | `UserAna2` interconnected |
| 26 | Show automatic preview in the standard view |
| 27 | Not used |
| 28 | `GrpErr.Value` |
| 29 | `RdyToStart.Value` |
| 30 | 1 = with position feedback messages (`Feature.Bit12`) |
| 31 | Internal error |

### Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 | Effective signal `1` of the message block connected via `EventTsIn` |
| 1 | Effective signal 2 of the message block connected via `EventTsIn` |
| 2 | Effective signal 3 of the message block connected via `EventTsIn` |
| 3 | Effective signal 4 of the message block connected via `EventTsIn` |
| 4 | Effective signal 5 of the message block connected via `EventTsIn` |
| 5 | Effective signal 6 of the message block connected via `EventTsIn` |
| 6 | Effective signal 7 of the message block connected via `EventTsIn` |
| 7 | Effective signal 8 of the message block connected via `EventTsIn` |
| 8 - 15 | Effective signal 9..16 of the message block connected via `EventTsIn` |
| 16 - 22 | Not used |
| 23 | Hidden bypass signal in Permit |
| 24 | Hidden bypass signal in interlock |

| Status bit | Parameter |
|---|---|
| 25 | Hidden bypass signal in Protect |
| 26 | `Feature2.Bit2`: Separate bypass signal |
| 27 | 1 = Input parameter `CtrlV0ChnST` is interconnected |
| 28 | 1 = Input parameter `CtrlV1ChnST` is interconnected |
| 29 | 1 = Input parameter `CtrlV2ChnST` is interconnected |
| 30 - 31 | Not used |

### See also

Vlv2WayL functions (Page 1653)

Vlv2WayL messaging (Page 1663)

Vlv2WayL block diagram (Page 1676)

Vlv2WayL error handling (Page 1660)

Vlv2WayL modes (Page 1651)

Resetting the block in case of interlocks or errors (Page 47)

## 8.9.2 Vlv2WayL modes

### Vlv2WayL operating modes

The block can be operated using the following modes:

- Local mode (Page 86)
- Automatic mode (Page 82)
- Manual mode (Page 82)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

### "Local mode"

You can find general information on "Local mode", switching modes and Bumpless switchover in the Local mode (Page 86) section.

Valve actions you can control in "local mode":

- Travel to neutral position (`Pos0Local` = 1)
- Moving to position 1 (`Pos1Local` = 1)
- Moving to position 2 (`Pos2Local` = 1).

A block operated in "local mode" is controlled either by "local" signals (input parameters `Pos0Local` = 1, `Pos1Local` = 1 and `Pos2Local` = 1) or by feedback signals (input parameters

`FdbV0`, `FdbV1`, `FdbV2` and `FdbP0`; if no position can be assigned, the last valid position is accepted). Configuration takes place via the input parameter `LocalSetting`.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and Bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 82) section.

Valve actions you can control in "automatic mode":

- Travel to neutral position (`Pos0Aut` = 1)
- Moving to position 1 (`Pos1Aut` = 1)
- Moving to position 2 (`Pos2Aut` = 1).

### "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the section Manual and automatic mode for motors, valves and dosers (Page 82).

Valve actions you can control in "manual mode":

- Travel to neutral position (`Pos0Man` = 1)
- Moving to position 1 (`Pos1Man` = 1)
- Moving to position 2 (`Pos2Man` = 1).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature2.Bit1` =0 and `Feature2.Bit1` =1 is as shown below:

| Output Parameters | **`Feature2.Bit1` =0** | **`Feature2.Bit1` =1** |
|---|---|---|
| FbkV0Out.ST | 16#80 | 16#60 |
| FbkV1Out.ST | 16#80 | 16#60 |
| FbkV2Out.ST | 16#80 | 16#60 |
| FbkP0Out.ST | 16#80 | 16#60 |
| FbkP1Out.ST | 16#80 | 16#60 |
| FbkP2Out.ST | 16#80 | 16#60 |
| Pos0.ST | 16#80 | 16#60 |
| Pos1.ST | 16#80 | 16#60 |
| Pos2.ST | 16#80 | 16#60 |

### See also

Vlv2WayL block diagram (Page 1676)

Vlv2WayL I/Os (Page 1664)

Vlv2WayL messaging (Page 1663)
Vlv2WayL error handling (Page 1660)
Vlv2WayL functions (Page 1653)
Description of Vlv2WayL (Page 1647)

## 8.9.3 Vlv2WayL functions

### Functions of Vlv2WayL

The functions for this block are listed below.

### Defining valve positions for individual valves

The control outputs for position 1 and position 2 can be selected individually with `DefPos1` and `DefPos2`:

| Route 1 or route 2 | Control outputs | | |
|---|---|---|---|
| **`DefPos1` or `DefPos2`** | **Valve `V0` (`CtrlV0`)** | **Valve `V1` (`CtrlV1`)** | **Valve `V2` (`CtrlV2`)** |
| 0 | closed | closed | closed |
| 1 | closed | closed | open |
| 2 | closed | open | closed |
| 3 | closed | open | open |
| 4 | open | closed | closed |
| 5 | open | closed | open |
| 6 | open | open | closed |
| 7 | open | open | open |

Position 0 is the neutral position (de-energized state) and cannot be configured. In position 0 all control outputs are de-energized (`CtrlVx` = 0).

### Output signal as a pulse signal or static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 57). In addition to the static control outputs `CtrlV0`, `CtrlV1`, `CtrlV2`, `Pos0Out`, `Pos1Out`, `Pos2Out`, the block also has pulse outputs `P_CtrlV0`, `P_CtrlV1`, `P_CtrlV2`, `P_CtrlP0`, `P_CtrlP1`, `P_CtrlP2` which are output depending on the static control outputs.

### Neutral position

This block provides the standard function Neutral position for motors, valves and controllers (Page 54). The neutral position (de-energized state) is set individually using parameters `SafeV0`, `SafeV1`, `SafeV2` for each valve (`CtrlV0`, `CtrlV1`, `CtrlV2`):

- `SafeVx` = 0 means that at `CtrlVx` = 0 the valve drive closes and at `CtrlVx` = 1 it opens (de-energized state is "closed")
- `SafeVx` = 1 means that at `CtrlVx` = 0 the valve drive opens and at `CtrlVx` = 1 it closes(de-energized state is "open")

### Specify warning times for control functions

This block provides the standard function Specifying warning times for control functions at motors and valves (Page 56). The warning signal is output before the valve moves into position 1 or position 2 . No warning signal is output for position 0 (neutral position).

You can generate warning signals when, for example, valves open. Warning signals can be generated in the following modes:

- Manual and automatic mode for motors, valves and dosers (Page 82) (Input parameter `WarnTiMan`)
- Manual and automatic mode for motors, valves and dosers (Page 82) (Input parameter `WarnTiAut`)

You specify the warning times in seconds using the input parameters `WarnTiMan` and `WarnTiAut`. If, for example, a valve opens then, this is displayed at the output parameter with `WarnAct` = 1. The valve then opens after the set warning time has expired and `WarnAct` then returns to 0.

A corresponding warning is not output if the warning times (`WarnTiMan` or `WarnTiAut`) are specified with a smaller value than the `SampleTime` parameter.

### Feedback monitoring

This block provides the standard function Monitoring the feedbacks (Page 104).

The monitoring of the feedback signal is dependent on `Feature.Bit12` "Position feedback signals are active (Page 179)".

### Valve feedback signals are active (`Feature.Bit12` = 0):

Startup characteristic monitoring is individually set up for every output signal `CtrlV0`, `CtrlV1`, `CtrlV2` via parameters `MonTiV0Dynamic`, `MonTiV1Dynamic` and `MonTiV2Dynamic` and is set for position 0 via `MonTiP0Dynamic`. The parameter `MonTiStatic` monitors compliance with the position.

The position feedback signals (FbkP1, FbkP2) are not used to monitor the control outputs.

**Note**

The monitoring function does not take into consideration the neutral positions (SafeV0, SafeV1, SafeV2), which means the feedback messages FbkV0, FbkV1, FbkV2 must correspond to the CtrlV0, CtrlV1, CtrlV2 controls (e.g., CtrlV0 = 1 means the feedback FbkV0 is monitored for "1").

FbkP0 must not occur for positions 1 or 2; at position 0, FbkV0, FbkV1 and FbkV2 must not occur.

If there are several feedback messages for position 0 (e.g. with a valve network), these must be combined using an upstream AND block at FbkP0.

### Position feedback signals are active (Feature.Bit12 = 1):

Startup characteristic monitoring is individually set up for the position feedback signals Pos1Out and Pos2Out via the parameters MonTiP1Dynamic and MonTiVP2Dynamic and for position 0 via MonTiP0Dynamic. The MonTiStatic parameter monitors compliance with the position.

The position feedback signals FbkP0, FbkP1 and FbkP2 are used to monitor the output signals Pos0Out, Pos1Out and Pos2Out.

**Note**

The separate valve feedback signals(FbkV0, FbkV1, FbkV2) are not used to monitor the control outputs.

### Disabling feedback

This block provides the standard function Disabling feedback for valves (Page 106). Feedback monitoring can be deactivated separately for each feedback with NoFbkV0, NoFbkV1, NoFbkV2, NoFbkP0, NoFbkP1, NoFbkP2.

### Suppressing messages using the MsgLock parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Forcing operating modes

This block provides the standard function Forcing operating modes (Page 45). The inputs Pos0Force, Pos1Force, Pos2Force force the block into position 0, position 1 or position 2.

### Simulating signals

This block provides the standard function Simulating signals (Page 64)

### Interlocks

This block provides the following interlocks:

- Activation enable
- Interlock without reset ("Interlock")
- Interlock with reset ("Protection")

Refer to the Interlocks (Page 107) section for more on this.

### Disabling interlocks

This block provides the standard function Disabling interlocks (Page 111).

### Resetting the block in case of interlocks or errors

This block provides the standard function Resetting the block in case of interlocks or errors (Page 47).

### External error (`FaultExt`), external control system fault (`CSF`)

This block provides the possibility to pass an external error via the `FaultExt` parameter or an external control system fault via the CSF parameter. See Vlv2WayL error handling (Page 1660)

### Group error

This block provides the standard function Outputting group errors (Page 130).

The following parameters are taken into consideration when forming the group error:

- `CSF`
- `MonDynV0` only with valve feedback signals (`Feature.Bit12` = 0)
- `MonDynV1` only with valve feedback signals (`Feature.Bit12` = 0)
- `MonDynV2` only with valve feedback signals (`Feature.Bit12` = 0)
- `MonDynP0`
- `MonDynP1`only with position feedback signals (`Feature.Bit12` = 1)
- `MonDynP2`only with position feedback signals (`Feature.Bit12` = 1)
- `MonStaV0`
- `MonStaV1`
- `MonStaV2`
- `MonStaP0`
- `FaultExt`

### Outputting a signal for start readiness

This block provides the standard function Outputting a signal for start readiness (Page 59).

### Forming the group status for interlocks

This block provides the standard function Forming the group status for interlock information (Page 113).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `FbkV0Out.ST` only with valve feedback signals (`Feature.Bit12` = 0)
- `FbkV1Out.ST` only with valve feedback signals (`Feature.Bit12` = 0)
- `FbkV2Out.ST` only with valve feedback signals (`Feature.Bit12` = 0)
- `FbkP0Out.ST`
- `FbkP1Out.ST`only with position feedback signals (`Feature.Bit12` = 1)
- `FbkP2Out.ST`only with position feedback signals (`Feature.Bit12` = 1)
- `LocalLi.ST`
- `Pos0Local.ST`
- `Pos1Local.ST`
- `Pos2Local.ST`
- `V0P0ChnST.ST`
- `V1P1ChnST.ST`
- `V2P2ChnST.ST`
- `Pos0Aut.ST` (only if `Feature2.Bit10` = 1)
- `Pos1Aut.ST` (only if `Feature2.Bit10` = 1)
- `Pos2Aut.ST` (only if `Feature2.Bit10` = 1)

### Considering bad quality of automatic commands or external values

If the `Feature2` bit Considering bad quality of automatic commands or external values (Page 200) is set to 1 and one of the following parameters has bad signal status (16#00 or 16#28), the block goes to the neutral position in the "Automatic" mode:

- `Pos0Aut.ST`
- `Pos1Aut.ST`
- `Pos2Aut.ST`

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 230).

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" |
| 1 | 1 = Operator can switch to "manual mode" |
| 2 | 1 = Operator can switch to "local mode" |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can switch to position 0 |
| 5 | 1 = Operator can switch to position 1 |
| 6 | 1 = Operator can switch to position 2 |
| 7 | 1 = Operator can reset the valve |
| 8 | 1 = Operator can define the monitoring time for startup |
| 9 | 1 = Operator can define the monitoring time for the end position |
| 10 | 1 = Operator can activate the monitoring time function (Bit 8 - 9) |
| 11 | Not used |
| 12 | 1 = Operator can activate the Release for maintenance function |
| 13 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Configurable reactions with the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 3 | Enabling resetting of commands for the control settings (Page 172) |
| 4 | Setting switch or button mode (Page 177) |
| 9 | Resetting via input signals in the event of interlocking (Protection) or errors (Page 174) |
| 10 | Exiting local mode (Page 189) |
| 11 | Activating the run time of feedback signals (Page 164) |
| 12 | Position feedback signals are active (Page 179) |
| 17 | Enabling bumpless switchover to automatic mode for valves, motors, and dosers (Page 184) |
| 18 | Activating error state for external process control error CSF (Page 162) |
| 20 | Disable calculation of impulse control in LocalSetting 2 and 4 (Page 200) |
| 21 | Enable bumpless switchover to "Automatic" mode for operator only (Page 183) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 27 | Interlock display with LocalSetting 2 or 4 (Page 189) |
| 30 | Set reset depending on the operating mode or the LiOp parameter (Page 174) |
| 31 | Activating reset of protection / error in manual mode (Page 175) |

In pushbutton operation (Bit 4 = 0) the automatic commands in "automatic mode" are latching, in other words `Pos0Aut`, `Pos1Aut`, `Pos2Aut` can be reset to 0 after switching to the selected position. In "manual" and "local" modes, however, the automatic commands are not saved and in the absence of automatic commands the position is tracked.

In switching mode (Bit 4 = 1) positions 1 and 2 are selected by static signals via inputs `Pos1Aut` and `Pos2Aut`. If inputs `Pos1Aut` and `Pos2Aut` are not set, the block switches to position 0. Control via `Pos0Aut` is not needed.

### Configurable reactions using the `Feature2` parameter

| Bit | Function |
|---|---|
| 1 | Changes signal status of outputs in OOS (Page 208) |
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 3 | Control priority in the event of invalid or not plausible input commands (Page 186) |
| 4 | Setting switch or button mode for local commands (Page 195) |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) |
| 8 | Forcing commands in the Local mode (Page 198) |
| 9 | Button mode with triggered local control commands (Page 209) |

| Bit | Function |
|---|---|
| 10 | Considering bad quality of automatic commands or external values (Page 200) |
| 11 | Suppress MsgLock and "Out of service" mode for a connected message block (Page 202) |
| 13 | Resetting the Pending Error in Local Mode 1 and 3 (Page 210) |

### Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 239).

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Button labels

This block provides the standard function Labeling of buttons and text (Page 235)

Instance-specific text can be configured for the following parameters:

- `Pos0Man`
- `Pos1Man`
- `Pos2Man`

### See also

Description of Vlv2WayL (Page 1647)

Vlv2WayL messaging (Page 1663)

Vlv2WayL I/Os (Page 1664)

Vlv2WayL block diagram (Page 1676)

Vlv2WayL modes (Page 1651)

EventTs functions (Page 2046)

## 8.9.4 Vlv2WayL error handling

### Vlv2WayL error handling

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Mode switchover error
- Control system fault (CSF)
- Invalid or not plausible input commands

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed. |
| 0 | There is no error. |
| 41 | The value for the `LocalSetting` I/O is not within the approved limit of 0 to 4. |
| 42 | `LocalSetting` = 0 and `LocalLi` = 1 |
| 51 | For `ModLiOp` = 1:<br>• `AutModLi` = 1 and `ManModLi` = 1<br>If "local mode" is enabled:<br>• `Pos0Local` = 1 and `Pos1Local` = 1<br>• `Pos0Local` = 1 and `Pos2Local` = 1<br>• `Pos1Local` = 1 and `Pos2Local` = 1<br>If "automatic mode" is enabled:<br>• `Pos0Aut` = 1 and `Pos1Aut` = 1<br>• `Pos0Aut` = 1 and `Pos2Aut` = 1<br>• `Pos1Aut` = 1 and `Pos2Aut` = 1<br>Generally:<br>• `Pos0Force` = 1 and `Pos1Force` = 1<br>• `Pos0Force` = 1 and `Pos2Force` = 1<br>• `Pos1Force` = 1 and `Pos2Force` = 1 |
| 52 | `LocalAct` = 1 and `LocalSetting` = 2 or 4 and `SimOn` = 1 |

### Mode switchover error

This error can be output by the block, see the section Error handling (Page 128).

### Invalid or not plausible input commands

This error can be output by the block, see the section Error handling (Page 128).

For the following invalid or not plausible input commands, the control output can be kept or switched to the neutral position. This depends on the function Control priority in the event of invalid or not plausible input commands (Page 186).

| **Operating mode** | **Invalid or not plausible input commands** | **Control reaction with `Feature2.Bit3` = 1** |
|---|---|---|
| Local: `Localsetting` = 1 or `Localsetting` = 3 | Pushbutton operation for local mode (`Feature2.Bit4` = 0):<br>`Pos1Local` = 1 and `Pos2Local` = 1 or<br>`Pos1Local` = 1 and `Pos0Local` = 1 or<br>`StopLocal` = 1 and `Pos2Local` = 1<br>Switching mode (`Feature2.Bit4` = 1):<br>`Pos1Local` = 1 and `Pos2Local` = 1 | Valve is set to Pos0. |
| Local: `Localsetting` = 1 or `Localsetting` = 3 and forcing | `Pos1Force` = 1 and `Pos2Force` = 1 or<br>`Pos1Force` = 1 and `Pos0Force` = 1 or<br>`Pos0Force` = 1 and `Pos2Force` = 1 | |
| Forcing and no "local mode" | `Pos1Force` = 1 and `Pos2Force` = 1 or<br>`Pos1Force` = 1 and `Pos0Force` = 1 or<br>`Pos0Force` = 1 and `Pos2Force` = 1 | |
| "Automatic mode" and no forcing | Pushbutton operation (`Feature.Bit4` = 0):<br>`Pos1Aut` = 1 and `Pos2Aut` = 1 or<br>`Pos1Aut` = 1 and `Pos0Aut` = 1 or<br>`Pos0Aut` = 1 and `Pos2Aut` = 1<br>Switching mode (`Feature.Bit4` = 1):<br>`Pos1Aut` = 1 and `Pos2Aut` = 1 | |
| "Manual mode" and no forcing | `Pos1Man` = 1 and `Pos0Man` = 1 or<br>`Pos1Man` = 1 and `Pos0Man` = 1 or<br>`Pos0Man` = 1 and `Pos2Man` = 1 | |

### Control system fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 128) section for more on this.

### See also

Vlv2WayL block diagram (Page 1676)

Vlv2WayL I/Os (Page 1664)

Vlv2WayL messaging (Page 1663)

Vlv2WayL functions (Page 1653)

Vlv2WayL modes (Page 1651)

Description of Vlv2WayL (Page 1647)

## 8.9.5 Vlv2WayL messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Instance-specific messages

### Process control fault

The following control system error messages can be output:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 1 | AS process control message - fault | $$BlockComment$$<br>Position 0 feedback error (neutral position) |
| | SIG 2 | AS process control message - fault | $$BlockComment$$<br>Position 1 or 2 feedback error |
| | SIG 3 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If it changes to CSF = 1, a process control fault is triggered (MsgEvId1, SIG 3).

### Instance-specific messages

You can use up to three instance-specific messages with this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External message 1 |
| | SIG 5 | AS process control message - fault | $$BlockComment$$<br>External message 2 |
| | SIG 6 | AS process control message - fault | $$BlockComment$$<br>External message 3 |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance `MsgEvId1`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | `ExtVa104` |
| 5 | `ExtVa105` |
| 6 | `ExtVa106` |
| 7 | `ExtVa107` |
| 8 | `ExtVa108` |
| 9 | Reserved |
| 10 | Reserved |

The associated values 4 ... 8 are allocated to the parameters `ExtVa104` ... `ExtVa108` , and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of Vlv2WayL (Page 1647)
Vlv2WayL functions (Page 1653)
Vlv2WayL I/Os (Page 1664)
Vlv2WayL block diagram (Page 1676)
Vlv2WayL error handling (Page 1660)
Vlv2WayL modes (Page 1651)

## 8.9.6 Vlv2WayL I/Os

I/Os of Vlv2WayL

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `AutModLi*` | 1= "Automatic mode" via:<br>Interconnection or SFC (controlled via `ModLiOp` = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `AutModOp*` | 1 = "Automatic mode" via operator (controlled by `ModLiOp` = 0) | BOOL | 0 |
| `BatchEn` | 1 = Enable allocation | BOOL | 0 |
| `BatchID` | Batch ID | DWORD | 16#00000000 |
| `BatchName` | Batch name | S7-String | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BypProt | 1 = Bypassing interlock in "local mode" and in "simulation" | BOOL | 0 |
| CSF | 1 = External error (control system fault) Error handling (Page 128) | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| DefPos1 | Output signal parameter setting for position 1 | INT | 3 |
| DefPos2 | Output signal parameter setting for position 2 | INT | 6 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| EventTsIn | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg3 | Binary input for freely selectable message 3 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| ExtVa104 | Associated value 4 for messages (MsgEvID1) | ANY | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ExtVa105 | Associated value 5 for messages (MsgEvID1) | ANY | |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| ExtVa108 | Associated value 8 for messages (MsgEvID1) | ANY | |
| FaultExt | 1 = External error<br>Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkP0 | 1 = Feedback for position 0 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#FF |
| FbkP1 | 1 = Feedback for position 1 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#FF |
| FbkP2 | 1 = Feedback for position 2 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#FF |
| FbkV0 | 1 = Feedback for control output CtrlV0 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#FF |
| FbkV1 | 1 = Feedback for control output CtrlV1 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#FF |
| FbkV2 | 1 = Feedback for control output CtrlV2 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#FF |
| Feature | I/O for additional functions (Page 1653) | STRUCT<br>• Bit:0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| Feature2 | I/O for additional functions (Page 1653) | STRUCT<br>• Bit:0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Intlock | 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared<br>1 = Interlock not activated | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| Intl_En | 1 = Interlock without reset (interlock, Intlock parameter) is active | BOOL | 1 |
| LocalLi | 1 = Activate "local mode" via plant signals | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| LocalOp* | 1 = "Local mode" via operator | BOOL | 0 |
| LocalSetting | Properties for the Local mode (Page 86) | INT | 0 |
| ManModLi* | 1 = "Manual mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| ManModOp* | 1 = "Manual mode" via OS operator (controlled by ModLiOp = 0) | BOOL | 1 |
| ModLiOp | Switchover of operating mode between:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Monitor | 1 = Feedback monitoring | BOOL | 1 |
| MonSafePos | 1 = Go to neutral position in the event of monitoring errors | BOOL | 1 |
| MonTiP0Dynamic* | Monitoring time for position 0 after operation in [s] | REAL | 3.0 |
| MonTiP1Dynamic* | Monitoring time for position 1 after operation in [s] | REAL | 3.0 |
| MonTiP2Dynamic* | Monitoring time for position 2 after operation in [s] | REAL | 3.0 |
| MonTiV0Dynamic* | Monitoring time for feedback errors FdbV0 after operation in [s] | REAL | 3.0 |
| MonTiV1Dynamic* | Monitoring time for feedback errors FdbV1 after operation in [s] | REAL | 3.0 |
| MonTiV2Dynamic* | Monitoring time for feedback errors FdbV2 after operation in [s] | REAL | 3.0 |
| MonTiStatic* | Monitoring time for feedback errors without operation in [s] | REAL | 3.0 |
| MS_RelOp* | 1= Release for maintenance via OS operator | BOOL | 0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NoFbkP0 | 1 = Feedback for position 0 not present | BOOL | 0 |
| NoFbkP1 | 1 = Feedback for position 1 not present | BOOL | 0 |
| NoFbkP2 | 1 = Feedback for position 2 not present | BOOL | 0 |
| NoFbkV0 | 1 = Feedback for control output CtrlV0 not present | BOOL | 0 |
| NoFbkV1 | 1 = Feedback for control output CtrlV1 not present | BOOL | 0 |
| NoFbkV2 | 1 = Feedback for control output CtrlV2 not present | BOOL | 0 |
| Occupied | 1 = In use by a batch | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 1653) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| Permit | 1 = Enable for opening / closing from neutral position<br>0 = Valve activation not enabled on OS | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| Perm_En | 1 = Activation enable (enable, Permit parameter) is active | BOOL | 1 |
| Pos0Aut* | 1 = Select position 0 in "automatic mode" | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Pos0Force | 1 = Force position 0 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Pos0Local | 1 = Select position 0 in "local mode" | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Pos0Man* | 1 = Select position 0 in "manual mode" | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Pos1Aut* | 1 = Select position 1 in "automatic mode" | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Pos1Force | 1 = Force position 1 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Pos1Local | 1 = Select position 1 in "local mode" | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Pos1Man* | 1 = Select position 1 in "manual mode" | BOOL | 0 |
| Pos2Aut* | 1 = Select position 2 in "automatic mode" | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Pos2Force | 1 = Force position 2 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Pos2Local | 1 = Select position 2 in "local mode" | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Pos2Man* | 1 = Select position 2 in "manual mode" | BOOL | 0 |
| Protect | 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block<br>1 = Protective interlocking not activated | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| Prot_En | 1 = Protective interlock (protection, Protect parameter) is active | BOOL | 1 |
| PulseWidth* | Pulse width of control signal [s] | REAL | 3.0 |
| RstLi* | 1 = Reset via interconnection | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | 1 = Reset via operator | BOOL | 0 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SafeV0 | Neutral position for valve V0 (CtrlV0):<br>1= Open<br>0 = Closed | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SafeV1 | Neutral position for valve V1 (CtrlV1):<br>1= Open<br>0 = Closed | BOOL | 0 |
| SafeV2 | Neutral position for valve V2 (CtrlV2):<br>1= Open<br>0 = Closed | BOOL | 0 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | |
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserAna1 | Analog auxiliary value 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UA1unit | Unit of measure for analog auxiliary value 1 | INT | 0 |
| UserAna2 | Analog auxiliary value 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UA2unit | Unit of measure for analog auxiliary value 2 | INT | 0 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |
| V0P0ChnST | Signal status of output channel for CtrlV0 or Pos0Out<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| V1P1ChnST | Signal status of output channel for CtrlV1 or Pos1Out<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| V2P2ChnST | Signal status of output channel for CtrlV2 or Pos2Out<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| WarnTiAut | Prewarning for valve movement into position 1 or position 2 in "automatic mode" in [s] | REAL | 0.0 |
| WarnTiMan | Prewarning for valve movement into position 1 or position 2 in "manual mode" in [s] | REAL | 0.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutAct | 1 = "Automatic mode" enabled<br>0 = "Manual mode" is enabled | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| AutoST | 1 = Worst signal status of automatic commands | BYTE | 16#80 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| CtrlV0 | Control output V0 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| CtrlV1 | Control output V1 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| CtrlV2 | Control output V2 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| CurrMonP0 | Remaining monitoring time P0 [s] | DINT | 0 |
| CurrMonP1 | Remaining monitoring time P1 [s] | DINT | 0 |
| CurrMonP2 | Remaining monitoring time P2 [s] | DINT | 0 |
| CurrMonV0 | Remaining monitoring time V0 [s] | DINT | 0 |
| CurrMonV1 | Remaining monitoring time V1 [s] | DINT | 0 |
| CurrMonV2 | Remaining monitoring time V2 [s] | DINT | 0 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see Vlv2WayL error handling (Page 1660) | INT | -1 |
| FaultExtOut | FaultExt as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkP0Out | Feedback from position 0 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| FbkP1Out | Feedback from position 1 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| FbkP2Out | Feedback from position 2 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| FbkV0Out | Control output feedback CtrlV0 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| FbkV1Out | Control output feedback CtrlV1 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| FbkV2Out | Control output feedback CtrlV2 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| GrpErr | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IntlockOut | Intlock as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalAct | 1 = "Local mode" enabled | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| LockAct | 1 = Interlock (Intlock, Permit or Protect) is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManAct | 1 = "Manual mode" enabled | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 1<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MonDynP0 | 1 = Feedback error for position 0 due to a control change | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| MonDynP1 | 1 = Feedback error for position 1 due to a control change | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| MonDynP2 | 1 = Feedback error for position 2 due to a control change | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| MonDynV0 | 1 = Feedback error for FdbV0 due to control change | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| MonDynV1 | 1 = Feedback error for FdbV1 due to control change | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| MonDynV2 | 1 = Feedback error for FdbV2 due to control change | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| MonStaP0 | 1 = Feedback error for position 0 due to an unexpected feedback change | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| MonStaP1 | 1 = Feedback error for position 1 due to an unexpected feedback change | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| MonStaP2 | 1 = Feedback error for position 2 due to an unexpected feedback change | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| MonStaV0 | 1 = Feedback error FdbV0 due to unexpected feedback change | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| MonStaV1 | 1 = Feedback error FdbV1 due to unexpected feedback change | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| MonStaV2 | 1 = Feedback error FdbV2 due to unexpected feedback change | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message acknowledgement status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MsgErr1 | Alarm error 1 (output ERROR of first ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| OosAct | 1 = Block is "out of service" | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| P_CtrlP0 | 1 = Pulse signal for moving the valve to position 0 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| P_CtrlP1 | 1 = Pulse signal for moving the valve to position 1 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| P_CtrlP2 | 1 = Pulse signal for moving the valve to position 2 | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| P_CtrlV0 | 1 = Pulse signal for moving the valve to route 0 (V0) | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| P_CtrlV1 | 1 = Pulse signal for moving the valve to route 1 (V1) | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| P_CtrlV2 | 1 = Pulse signal for moving the valve to route 2 (V2) | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| P_Rst | 1= Pulse output for reset<br>The parameter persists for one cycle after a reset. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PermitOut | Permit as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Pos0 | 1 = Pos0 is reached | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Pos1 | 1 = Pos1 is reached | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Pos2 | 1 = Pos2 is reached | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Pos0Out | 1 = Position 0 is active | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Pos1Out | 1 = Position 1 is active | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Pos2Out | 1 = Position 2 is active | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| ProtectOut | Protect as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToReset | 1 = Ready for reset via RstLi input or commands in "local", "automatic", or "manual mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToStart | 1 = Active start readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 1647) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1647) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1647) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 1647) | DWORD | 16#00000000 |
| UA1unitOut | UA1 as output | INT | 0 |
| UA2unitOut | UA2 as output | INT | 0 |
| UserAna1Out | UserAna1 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| `UserAna2Out` | `UserAna2` as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| `WarnAct` | 1 = Prewarning for valve movement to position 1 or position 2 enabled (parameter `WarnTiAut` and `WarnTiMan`) | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |

**See also**

Vlv2WayL messaging (Page 1663)
Vlv2WayL block diagram (Page 1676)
Vlv2WayL modes (Page 1651)
Error handling (Page 128)

## 8.9.7 Vlv2WayL block diagram

### Vlv2WayL block diagram

A block diagram is not provided for this block.

**See also**

Vlv2WayL I/Os (Page 1664)
Vlv2WayL messaging (Page 1663)
Vlv2WayL error handling (Page 1660)
Vlv2WayL functions (Page 1653)
Vlv2WayL modes (Page 1651)
Description of Vlv2WayL (Page 1647)

## 8.9.8 Operator control and monitoring

### 8.9.8.1 Vlv2WayL views

#### Views of the Vlv2WayL block

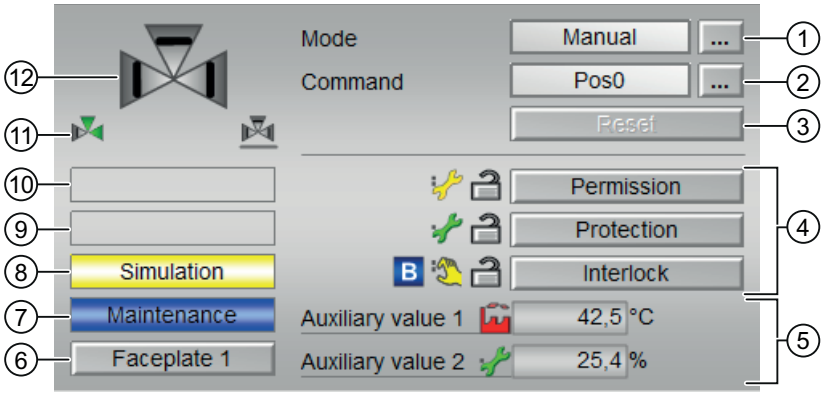
The block Vlv2WayL provides the following views:

- Vlv2WayL standard view (Page 1677)
- Alarm view (Page 336)
- Trend view (Page 341)
- Vlv2WayL parameter view (Page 1681)
- Vlv2WayL preview (Page 1683)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for Vlv2WayL (Page 1688)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 8.9.8.2 Vlv2WayL standard view

#### Vlv2WayL standard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual and automatic mode for motors, valves and dosers (Page 82)
- Manual and automatic mode for motors, valves and dosers (Page 82)

- Local mode (Page 86)
- Out of service (Page 77)

Refer to section Switching operating states and operating modes (Page 289) for information on switching the operating mode.

### (2) Selecting the position for 2-way valve

This area shows you the default operating state for the valve. The following states can be shown and executed here:

- "Pos0"
- "Pos1"
- "Pos2"

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

If the text is configured for these commands, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (3) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in the section Resetting the block in case of interlocks or errors (Page 47).

### (4) Operating range for the interlock functions of the block

This display is only visible when the corresponding block input is connected.

You can use this button to control the interlock functions of the block. You can find additional information on this in the section Interlocking functions (Page 107).

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 113)), for example:

- Signal status (see Forming and outputting the signal status for technologic blocks (Page 117)), for example:

- Bypass information (see Forming the group status for interlock information (Page 113)):

If the text is configured for the interlocks, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (5) Display of auxiliary values

This display is only visible when the corresponding block input is connected.

You can use this area to display two auxiliary values that have been configured in the engineering system (ES). You can find additional information on this in the section Displaying auxiliary values (Page 239).

### (6) Navigation button for switching to the standard view of any faceplate

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234).

### (7) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this is available in section Release for maintenance (Page 70) Display area for block states.

### (8) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"
- "Delay"

You will find more detailed information on this in the sections Simulating signals (Page 64) and Display of delay times (Page 288).

### (9) Display area for block states

This area provides additional information on the operating state of the block:

- "External error"
- "End position error"
- "Control error"
- "Internal error"
- "Changeover error"

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 104) , Error handling (Page 128) (section "Invalid input signals" and "Mode switchover error") and Motor protection function (Page 106).

### (10) Display area for block states

This area provides additional information on the operating state of the block:

- "Force Pos0"
- "Force Pos1"
- "Force Pos2"
- "Request 0/1": Reset is expected but reset is not possible from faceplate. For more details, refer:Reset of errors during Request 0/1 (Page 382)

You can find additional information on this in the section Forcing operating modes (Page 45).

### (11) Automatic preview

This display is only visible in "manual mode", in "local mode", or with a reset request in "automatic mode", when the current output signals are not identical to the control in "automatic mode".

The display shows what state the valve would assume if you switched from "manual" or "local" mode to "automatic mode", or performed a reset to "automatic mode".

### (12) Status display of the valve

You can find additional information on this in the section Block icon for Vlv2WayL (Page 1688).

### (13) Neutral position of the valve

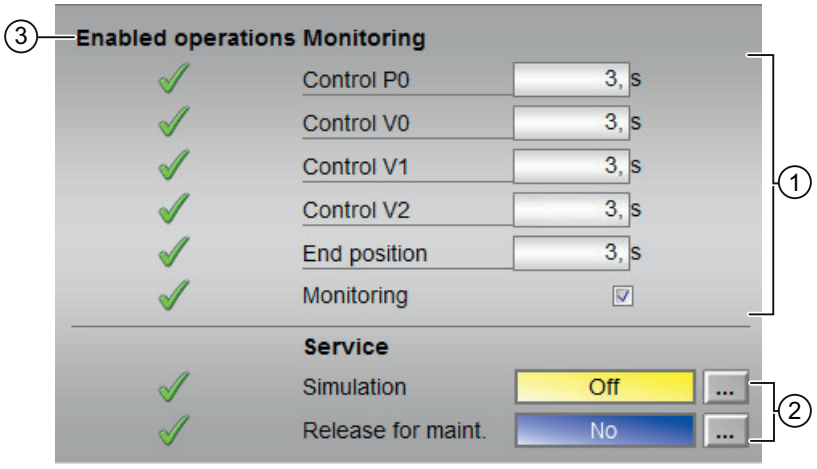
The neutral position of the valve is shown here.

If the neutral position of the valve is "Closed" (SafePos = 0), a gray valve is shown.

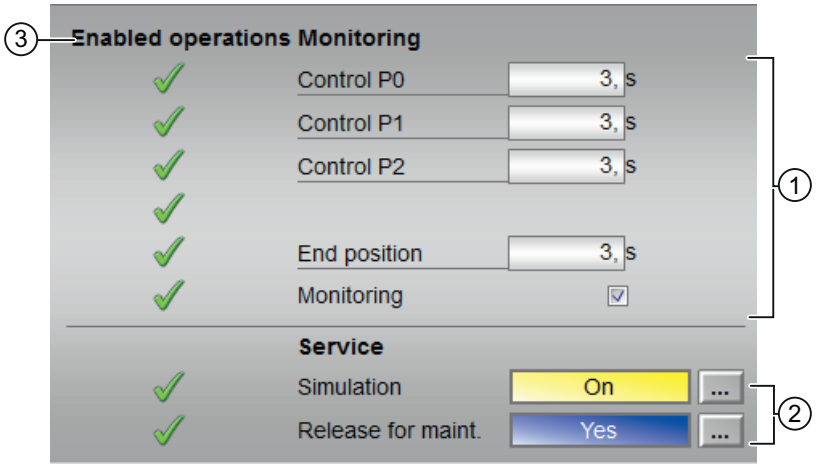
If the neutral position of the valve is "Open" (SafePos = 0), a green valve is shown.

### 8.9.8.3 Vlv2WayL parameter view

#### Parameter view of Vlv2WayL




Two-way valve with separate valve feedback signals (`Feature.Bit12` = 0)




Two-way valve with position feedback signals (`Feature.Bit12` = 1)

#### (1) Monitoring

In this area, you change parameters and therefore influence the valve. Refer to the Changing values (Page 291) section for more on this.

You can influence the following parameters:

- "Control P0": Monitoring time while "opening"/"closing" the valve
- "Control P1": Monitoring time while "opening"/"closing" the valve
- "Control P2": Monitoring time while "opening"/"closing" the valve

- "Control V0": Monitoring time while "opening"/"closing" the valve
- "Control V1": Monitoring time while "opening"/"closing" the valve
- "Control V2": Monitoring time while "opening"/"closing" the valve
- "End position": Monitoring time for maintaining the valve position

**Enable monitoring**

You can enable monitoring by selecting the check box (☑)

You can find additional information on this in the section Monitoring the feedbacks (Page 104).

### (2) Service

You can select the following functions in this area:

- "Simulation"
- "Release for maintenance"

Refer to the Switching operating states and operating modes (Page 289) section for more on this.

### (3) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

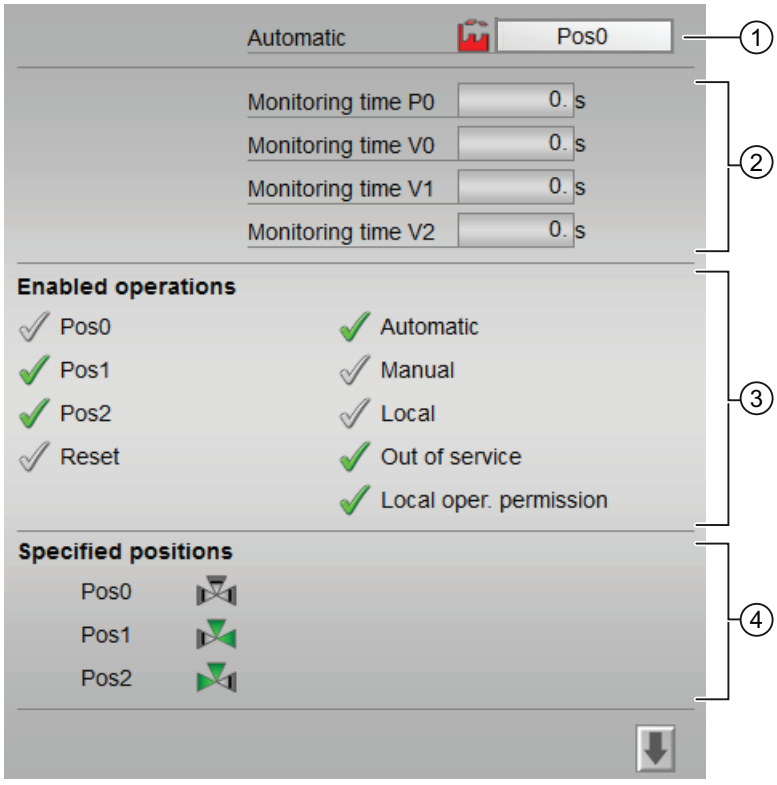
Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

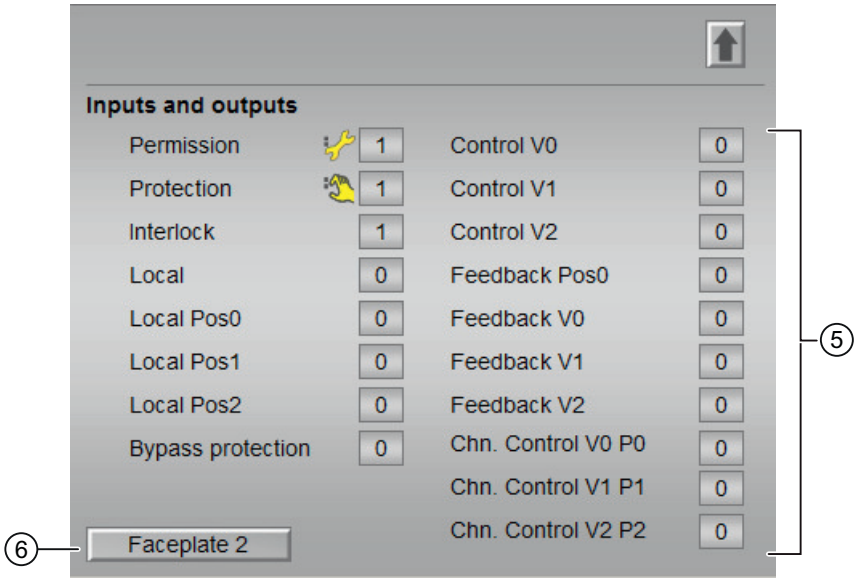
### 8.9.8.4 Vlv2WayL preview

#### Preview of Vlv2WayL

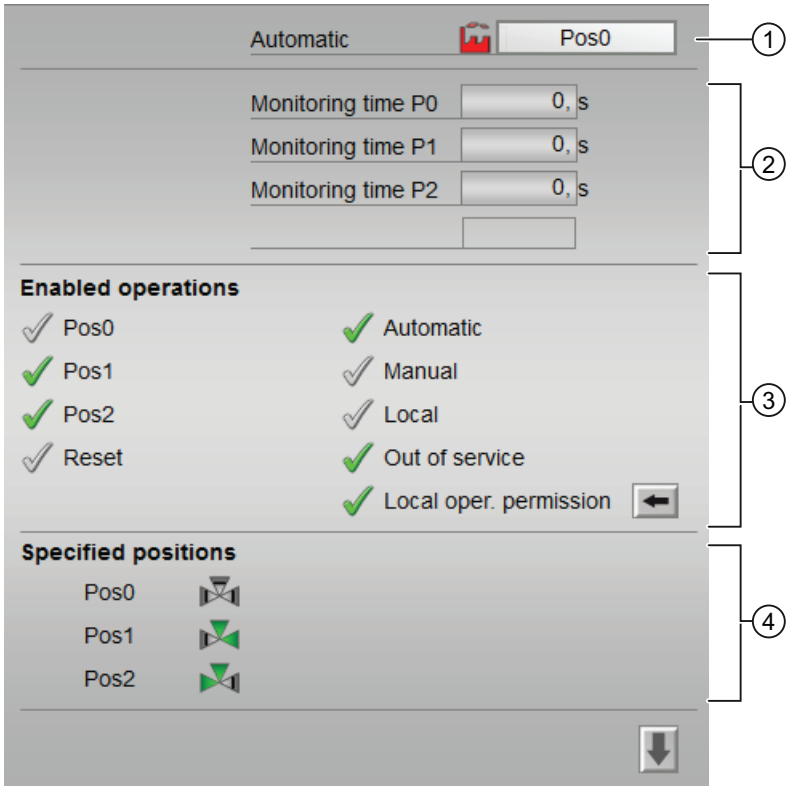
Two-way valve with single value feedback signals (`Feature.Bit12` = 0)



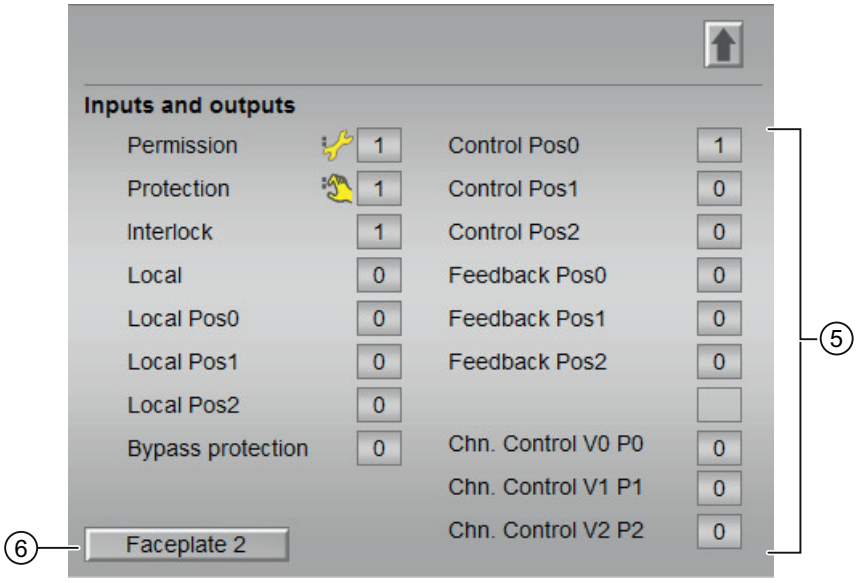

The preview has an upper half and a lower half. You can change between the two halves with the arrow keys.




Two-way valve with position feedback signals (`Feature.Bit12` = 1)




The preview has an upper half and a lower half. You can change between the two halves with the arrow keys.




## (1) Automatic preview

This area shows you the block status after it has switched from the "Manual" mode or "Local" mode to the "Automatic" mode.

If the block is in "Automatic" mode, the current block state is displayed.

**Note**

- If the valve is closed and interlock locking is active (`SafePos`=0 Close), the automatic preview does not show the current status "Closed", instead it shows the active automatic command "Open".
- If the valve is in the end position and the automatic stop command is active, the automatic preview does not show the current status "Closed", but the active automatic command "Stop".

This area also displays the worst signal status of the following automatic commands:

- `Pos0Aut`
- `Pos1Aut`
- `Pos2Aut`

## (2) Preview area

The following enabled operations are shown here:

- "Monitoring time `P0`": Display of the remaining monitoring time `P0` (`Feature.Bit12` = 1)
- "Monitoring time `P1`": Display of the remaining monitoring time `P1` (`Feature.Bit12` = 1)
- "Monitoring time `P2`": Display of the remaining monitoring time `P2` (`Feature.Bit12` = 1)
- "Monitoring time `V0`": Display of the remaining monitoring time `V0` (`Feature.Bit12` = 0)

- "Monitoring time `V1`": Display of the remaining monitoring time `V1` (`Feature.Bit12` = 0)
- "Monitoring time `V2`": Display of the remaining monitoring time `V2` (`Feature.Bit12` = 0)

### (3) Enable operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "Pos0": You can set the valve to position 0.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Pos1": You can set the valve to position 1.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Pos2": You can set the valve to position 2.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Reset": You can reset the valve if errors occur.
- "Automatic": You can switch to "automatic mode".
- "Manual": You can switch to "manual mode".
- "Local": You can switch to "local mode".
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operator permission": Use the ← button to switch to the standard view of the OpStations block. You can find additional information on this in the section Operator control permissions (Page 285).

### (4) Specified Position

Preview of the valve positions, as configured in the engineering system (ES).

### (5) Displaying current control signals

This area shows the most important parameters for this block with the current selection:

- "Permission":
  This display is only visible when the corresponding block input is connected.
  - 0 = Valve activation not enabled on OS
  - 1 = Enable for "starting"/"stopping" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection":
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock":
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Local": 1 = Control signal for "Local mode" (LocalLi) is active
- "Local Pos0": 1 = Block was set to position 0 in "local mode"
- "Local Pos1": 1 = Block was set to position 1 in "local mode"
- "Local Pos2": 1 = Block was set to position 2 in "local mode"
- "Interlock deact.":
  - 0 = Bypass disabled
  - 1 = Bypassing interlock in "local mode" and in "simulation"

Control and feedback signals with `Feature.Bit12` = 0

- "Control Pos0": 1 = Control signal for the position 0
- "Control V0": 1 = Control signal for the valve 0
- "Control V1": 1 = Control signal for the valve 1
- "Control V2": 1 = Control signal for the valve 2
- "Feedback Pos0": 1 = Valve is in position 0
- "Feedback V0": 1 = Feedback if valve 0 was opened
- "Feedback V1": 1 = Feedback if valve 1 was opened
- "Feedback V2": 1 = Feedback if valve 2 was opened

Control and feedback signals with `Feature.Bit12` = 1

- "Control Pos0": 1 = Control signal for the position 0
- "Control Pos1": 1 = Control signal for the position 1
- "Control Pos2": 1 = Control signal for the position 2
- "Feedback Pos0": 1 = Valve is in position 0
- "Feedback Pos1": 1 = Valve is in position 1
- "Feedback Pos2": 1 = Valve is in position 2

Control signals of the output channel block

- "Channel control V0 P0": Control signal for V0 or P0 of the output channel block
- "Channel control V1 P1": Control signal for V1 or P1 of the output channel block
- "Channel control V2 P2": Control signal for V2 or P2 of the output channel block

**(6) Navigation button for switching to the standard view of any faceplate**

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

For more information, see section Opening additional faceplates (Page 234).

### 8.9.8.5 Block icon for Vlv2WayL

**Properties of the Vlv2WayLblock icon**

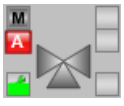
A variety of block icons are available with the following functions:

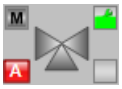
- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as the control system fault
- Operating modes
- Signal status, release for maintenance
- Forcing states
- Displays for bypassing interlocks
- Interlocks
- Memo display
- Valve status display

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | |
|  | 4 | |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | |
|  | 6 | |
|  | 7 | |
|  | 8 | |
|  | 9 | |

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 10 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

## Valve status display

The following valve states are shown here:

| Icon | Meaning |
|---|---|
|  | Valve open |
|  | Error at valve |
|  | Valve is opening |
|  | Valve closed |
|  | Valve is closing |

# 8.10 VlvL - Valve (Large)

## 8.10.1 Description of VlvL

### Object name (type + number) and family

Type + number: FB 1899

Family: Drives

### Area of application for VlvL

The block is used for the following applications:

- Controlling a valve in two positions ("open"/"closed") with adjustable neutral position

**Note**

This block is also available as a small block. A comparison of the VlvL and VlvS blocks is available in the section: VlvL compared to VlvS (Page 1372)

### How it works

The valve is opened or closed by a control signal. The signal 0 corresponds to the de-energized state (neutral position) of the valve.

The control is monitored by the "open"/"close" (feedback) signals. Missing feedback can be derived from the control in the block.

Various inputs are available for control purposes. The next sections provide more detailed information on configuration, operating principles, visualization and operation.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

For the VlvL block, the Advanced Process Library contains a template for process tag types as an example with an application scenario for this block.

Example of process tag types:

- Valve (Valve_Lean) (Page 2865)

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section VlvL I/Os (Page 1707).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `AutoAct.Value` |
| 6 | `LocalAct.Value` |
| 7 | 0: Open padlock in the block icon<br>1: Closed padlock in the block icon |
| 8 | "Open"/"Closed" command (1 = "Open") |
| 9 | `FbkOpenOut.Value` |
| 10 | `FbkCloseOut.Value` |
| 11 | Feedback error without control change |
| 12 | Feedback error due to control change |
| 13 | `BypProt` |
| 14 | Invalid signal status |
| 15 | Mode switchover error |
| 16 | 1 = `Intlock` is active |
| 17 | 1 = `Permit` is active |
| 18 | 1 = `Protect` is active |
| 19 | Display „Forced open“ |
| 20 | Display „Forced close“ |
| 21 | `Force` |
| 22 | Automatic preview (1 = "Open") |
| 23 | Bumpless switchover to "automatic mode" enabled |
| 24 | `SafePos` |
| 25 | `UserAna1` interconnected |
| 26 | `UserAna2` interconnected |
| 27 | `WarnAct.Value` |
| 28 | For the Error status display in the Closed valve |
| 29 | For the Error status display in the Opened valve |
| 30 | External error generated by FaultExt or external control system fault from CSF with set `Feature.Bit18` Activating error state for external process control error CSF (Page 162) |
| 31 | 1 = Input parameter `CtrlChnST` is interconnected |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `MsgLock` |
| 1 | Not used |
| 2 | Display for interlocks in block icon |
| 3 - 11 | Not used |
| 12 | Resynchronization preview for Open |
| 13 | Resynchronization preview for Close |
| 14 | Resynchronization: Display preview |
| 15 | Resynchronization: Display small preview symbol in standard view |
| 16 | 1 = Input parameter `FbkClose` is connected |
| 17 | 1 = Input parameter `FbkOpen` is connected |
| 18 | Reset request in automatic |
| 19 | 1 = No impact of input signals on "local mode" with `LocalSetting` = 0, `LocalSetting` = 2, `LocalSetting` = 4 and OOS mode. |
| 20 | 1 = Valve open |
| 21 | 1 = Valve closed |
| 22 | 1 = Valve opens |
| 23 | 1 = Valve closes |
| 24 | Resynchronization with state "No Cmd" |
| 25 - 29 | Not used |
| 30 | Bypass information for display in block icon |
| 31 | `MS_RelOp` |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 | Effective signal 1 of the message block connected via `EventTsIn` |
| 1 | Effective signal 2 of the message block connected via `EventTsIn` |
| 2 | Effective signal 3 of the message block connected via `EventTsIn` |
| 3 | Effective signal 4 of the message block connected via `EventTsIn` |
| 4 | Effective signal 5 of the message block connected via `EventTsIn` |
| 5 | Effective signal 6 of the message block connected via `EventTsIn` |
| 6 | Effective signal 7 of the message block connected via `EventTsIn` |
| 7 | Effective signal 8 of the message block connected via `EventTsIn` |
| 8 | "Interlock" button is enabled |
| 9 | "Permission" button is enabled |
| 10 | "Protection" button is enabled |
| 11 | Hidden bypass signal in Permit |
| 12 | Hidden bypass signal in interlock |
| 13 | Hidden bypass signal in Protect |
| 14 | `Feature2.Bit2`: Separate bypass signal |
| 15 | Current monitoring time is visible |

| Status bit | Parameter |
|---|---|
| 16 | 1 = " Permit neutral pos." button is enabled |
| 17 | 1 = Permission SafePos is active |
| 18 | Hidden bypass signal in Permission SafePos |
| 19 - 25 | Not used |
| 26 | Show automatic preview in the standard view |
| 27 | Not used |
| 28 | `GrpErr.Value` |
| 29 | `RdyToStart.Value` |
| 30 | Not used |
| 31 | Internal error |

### Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 - 7 | Effective signal 9..16 of the message block connected via `EventTsIn` |
| 8 - 31 | Not used |

### See also

VlvL functions (Page 1696)

VlvL messaging (Page 1705)

Overview of the modes (Page 76)

VlvL block diagram (Page 1715)

VlvL error handling (Page 1703)

VlvL modes (Page 1694)

Resetting the block in case of interlocks or errors (Page 47)

## 8.10.2 VlvL modes

### VlvL operating modes

The block supports all standard modes:

- Local mode (Page 86)
- Automatic mode (Page 82)
- Manual mode (Page 82)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

### "Local mode"

You can find general information on "Local mode", switching modes and Bumpless switchover in the Local mode (Page 86) section.

Valve actions you can control in "local mode":

- "Open" (`OpenLocal` = 1)
- "Close" (`CloseLocal` = 1)

A block operated in "local mode" is controlled either by "local" signals or by feedback signals (input parameters `FbkOpen`, `FbkClose` or `FbkStop`; if no position can be assigned, the last valid position is accepted). Configuration takes place via the input parameter `LocalSetting`.

`LocalSetting` =2,4: The control outputs are tracked to the feedbacks.

All two feedbacks are available and plausible means that always only one feedback is active. The corresponding control output is tracked:

```
Ctrl = FbkOpen XOR SafePos
```

```
StopCtrl = FbkClose XOR SafePos
```

When both feedbacks are false then the drive is between of two positions. In this case no monitoring is started. When two feedbacks are active at the same time, then they are not plausible. In this case the static monitoring counter starts. In both cases "all feedbacks are reset" or "not plausible feedbacks" with `Feature2Bit21` =1 all the control outputs are set to false (No command state) else control outputs will be hold.

If one feedback is available it is always plausible.

| NoFbkClose =0 | NoFbkOpen =0 |
|---|---|
| `Ctrl = NOT (FbkClose XOR SafePos)`<br>`StopCtrl = FbkClose XOR SafePos` | `Ctrl = FbkOpen XOR SafePos`<br>`StopCtrl = NOT (FbkOpen XOR SafePos)` |

When all feedbacks are not available the control outputs will be hold.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and Bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 82) section.

Valve actions you can control in "automatic mode":

- "Open" (`OpenAut` = 1)
- "Close" (`CloseAut` = 1)

### "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the section Manual and automatic mode for motors, valves and dosers (Page 82).

Valve actions you can control in "manual mode":

- "Open" (`OpenMan` = 1)
- "Close" (`CloseMan` = 1)

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature2.Bit1` =0 and `Feature2.Bit1` =1 is as shown below:

| Output Parameters | **`Feature2.Bit1` =0** | **`Feature2.Bit1` =1** |
|---|---|---|
| FbkOpenOut.ST | 16#80 | 16#60 |
| FbkCloseOut.ST | 16#80 | 16#60 |
| Opened.ST | 16#80 | 16#60 |
| Closed.ST | 16#80 | 16#60 |
| Opening.ST | 16#80 | 16#60 |
| Closing.ST | 16#80 | 16#60 |

### See also

Description of VlvL (Page 1691)

VlvL block diagram (Page 1715)

VlvL I/Os (Page 1707)

VlvL messaging (Page 1705)

VlvL error handling (Page 1703)

VlvL functions (Page 1696)

## 8.10.3 VlvL functions

### Functions of VlvL

The functions for this block are listed below.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" |
| 1 | 1 = Operator can switch to "manual mode" |
| 2 | 1 = Operator can switch to "local mode" |

| Bit | Function |
|---|---|
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can open the valve |
| 5 | 1 = Operator can close the valve |
| 6 | 1 = Operator can reset the valve |
| 7 | 1 = Operator can define the monitoring time for startup |
| 8 | 1 = Operator can define the monitoring time for the end position |
| 9 | 1 = Operator can activate the monitoring time function (Bit 7 - 8) |
| 10 | 1 = Operator can enable function simulation |
| 11 | 1 = Operator can activate the Release for maintenance function |
| 12 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Interlocks

This block provides the following interlocks:

- Activation enable
- Interlock without reset ("Interlock")
- Interlock with reset ("Protection")
- Permission for safe position (“Permit neutral pos.”)

Refer to the Interlocks (Page 107) section for more on this.

### Disabling interlocks

This block provides the standard function Disabling interlocks (Page 111).

### Resetting the block in case of interlocks

This block provides the standard function Resetting the block in case of interlocks or errors (Page 47).

### External error (`FaultExt`), external control system fault (`CSF`)

This block provides the possibility to pass an external error via the `FaultExt` parameter or an external control system fault via the CSF parameter. See VlvL error handling (Page 1703)

### Group error

This block provides the standard function Outputting group errors (Page 130).

The following parameters are taken into consideration when forming the group error:

- `CSF`
- `MonDynErr`
- `MonStaErr`
- `FaultExt`

### Outputting a signal for start readiness

This block provides the standard function Outputting a signal for start readiness (Page 59).

### Forming the group status for interlocks

This block provides the standard function Forming the group status for interlock information (Page 113).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status ST_Worst for the block is formed from the following parameters:

- `FbkOpenOut.ST`
- `FbkCloseOut.ST`
- `LocalLi.ST`
- `OpenLocal.ST`
- `CloseLocal.ST`
- `CtrlChn.ST`
- `OpenAut.ST` (only if `Feature2.Bit10` = 1)
- `CloseAut.ST` (only if `Feature2.Bit10` = 1)

### Considering bad quality of automatic commands or external values

If the `Feature2` bit Considering bad quality of automatic commands or external values (Page 200) is set to 1 and one of the following parameters has bad signal status (16#00 or 16#28), the block goes to the neutral position in the "Automatic" mode:

- `OpenAut.ST`
- `CloseAut.ST`

### Forcing operating modes

This block provides the standard function Forcing operating modes (Page 45).
Inputs `OpenForce` and `CloseForce` force the block to open or close.

### Feedback monitoring

This block provides the standard function Monitoring the feedbacks (Page 104). Startup characteristics are monitored by setting parameter`MonTiDynamic`. The parameter `MonTiStatic` monitors compliance with the position.

### Disabling feedback

This block provides the standard function for Disabling feedback for valve and motors (Page 106).

Feedback monitoring can be deactivated separately for each feedback with `NoFbkOpen` or `NoFbkClose` as required.

The output signals of feedback are tracked by the control and the other available feedbacks:

| NoFbkOpen | NoFbkClose | Comment |
|---|---|---|
| 0 | 0 | **Default:**<br>`FbkOpenOut = FbkOpen;`<br>`FbkCloseOut = FbkClose;` |
| 0 | 1 | `FbkOpenOut = FbkOpen;`<br>`FbkCloseOut = (StopCtrl XOR SafePos) AND NOT FbkOpen;`<br>**Local mode with LocalSetting =2/4:**<br>`FbkOpenOut = FbkOpen;`<br>`FbkCloseOut = NOT FbkOpen;` |
| 1 | 0 | `FbkOpenOut = (Ctrl XOR SafePos) AND NOT FbkClose;`<br>`FbkCloseOut = FbkClose;`<br>**Local mode with LocalSetting =2/4:**<br>`FbkOpenOut = NOT FbkClose;`<br>`FbkCloseOut = FbkClose;` |
| 1 | 1 | `FbkOpenOut = Ctrl XOR SafePos;`<br>`FbkCloseOut = StopCtrl XOR SafePos;` |

When all feedbacks are not available, the position is on hold after a switch to local mode with `LocalSetting` =2/4.

### Disabling feedback

This block provides the standard function Disabling feedback for valves (Page 106). Feedback monitoring can be deactivated separately for each feedback with `NoFbkOpen` or `NoFbkClose` as required.

### Using StopCtrl with mode "No Cmd"

This block provides the standard function Control with output StopCtrl and state “No Cmd” (Page 214)

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Specify warning times for control functions

This block provides the standard function Specifying warning times for control functions at motors and valves (Page 56). The warning signal is output before the valve moves away from the neutral position. No signal is output for movement to the neutral position.

You can generate warning signals when, for example, valves open. Warning signals can be generated in the following modes:

- Manual and automatic mode for motors, valves and dosers (Page 82) (Input parameter `WarnTiMan`)
- Description of VlvL (Page 1691) (Input parameter `WarnTiAut`)

You specify the warning times in seconds using the input parameters `WarnTiMan` and `WarnTiAut`. If, for example, a valve opens then, this is displayed at the output parameter with `WarnAct` = 1. The valve then opens after the set warning time has expired and `WarnAct` then returns to 0.

A corresponding warning is not output if the warning times (`WarnTiMan` or `WarnTiAut`) are specified with a smaller value than the `SampleTime` parameter.

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

### Neutral position

This block provides the standard function Neutral position for motors, valves and controllers (Page 54). The neutral position (de-energized state) is set using the `SafePos` parameter.

- `SafePos` = 0 means that at `Ctrl` = 0 the valve drive closes and at `Ctrl` = 1 it opens (de-energized state is "closed")
- `SafePos` = 1 means that at `Ctrl` = 0 the valve drive opens and at `Ctrl` = 1 it closes (de-energized state is "open")

### Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 230).

## Resynchronization

This block provides the standard function Resynchronization (Page 258).

Resynchronization is switched on over feature bits:

| | |
|---|---|
| Feature2 Bit17 =1 | Resynchronization after CPU startup |
| Feature2 Bit18 =1 | Resynchronization after Simulation or after Out of service |
| Feature2 Bit19 =1 | Resynchronization after control system fault |
| Feature2 Bit20 =1 | Resynchronization with 0->1 trigger at StateTrkOn input |

If there are no restrictions the control outputs are tracked as follow:

**NoFbkOpen =0, NoFbkClose =0 (default):**

Plausible feedback positions (FbkOpen =1, FbkClose =0 or FbkOpen =0, FbkClose =1) are tracked as control position.

**SafePos =0:** Ctrl := FbkOpen; StopCtrl := FbkClose;

**SafePos =1:** Ctrl := FbkClose; StopCtrl := FbkOpen;

In case if there are no plausible feedback positions (FbkOpen =0, FbkClose =0 or FbkOpen =1, FbkClose =1) the control position depends on Feature2 Bit21.

- Feature2 Bit21 =0 (default): Control position is not changed.
- Feature2 Bit21 =1: Control position becomes "No Cmd" Ctrl := 0; StopCtrl := 0;

**NoFbkOpen =1, NoFbkClose =0:**

**SafePos =0:** Ctrl := NOT FbkClose; StopCtrl := FbkClose;

**SafePos =1:** Ctrl := FbkClose; StopCtrl := NOT FbkClose;

**NoFbkOpen =0, NoFbkClose =1:**

**SafePos =0:** Ctrl := NOT FbkOpen; StopCtrl := FbkOpen;

**SafePos =1:** Ctrl := FbkOpen; StopCtrl := NOT FbkOpen;

With other combinations **NoFbkOpen =1**, **NoFbkClose =1** the controls are not changed

## Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 3 | Enabling resetting of commands for the control settings (Page 172) |
| 4 | Setting switch or button mode (Page 177) |
| 9 | Resetting via input signals in the event of interlocking (Protection) or errors (Page 174) |

| Bit | Function |
|---|---|
| 10 | Exiting local mode (Page 189) |
| 11 | Activating the run time of feedback signals (Page 164) |
| 17 | Enabling bumpless switchover to automatic mode for valves, motors, and dosers (Page 184) |
| 18 | Activating error state for external process control error CSF (Page 162) |
| 20 | Disable calculation of impulse control in LocalSetting 2 and 4 (Page 200) |
| 21 | Enable bumpless switchover to "Automatic" mode for operator only (Page 183) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 27 | Interlock display with LocalSetting 2 or 4 (Page 189) |
| 30 | Set reset depending on the operating mode or the LiOp parameter (Page 174) |
| 31 | Activating reset of protection / error in manual mode (Page 175) |

In pushbutton operation (Bit 4 = 0) the automatic commands in "automatic" mode are latching, in other words `OpenAut`, `CloseAut` can be reset to zero after changing the control. In "manual" and "local" modes, however, the automatic commands are not saved and in the absence of automatic commands the automatic control is tracked.

In switching mode (Bit 4 = 1), the control is selected with the static signal `OpenAut`. If input `OpenAut` is not set the valve is closed. Control via `CloseAut` is not needed. If the "Activate command reset for control" function (Bit 3 = 1) is also activated, the `OpenAut` input is reset to the neutral position after evaluation in the block.

### Configurable reactions using the `Feature2` parameter

| Bit | Function |
|---|---|
| 1 | Changes signal status of outputs in OOS (Page 208) |
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 3 | Control priority in the event of invalid or not plausible input commands (Page 186) |
| 4 | Setting switch or button mode for local commands (Page 195) |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) |
| 8 | Forcing commands in the Local mode (Page 198) |
| 9 | Button mode with triggered local control commands (Page 209) |
| 10 | Considering bad quality of automatic commands or external values (Page 200) |
| 11 | Suppress MsgLock and “Out of service” mode for a connected message block (Page 202) |
| 13 | Resetting the Pending Error in Local Mode 1 and 3 (Page 210) |
| 17 | Resynchronization after startup (Page 213) |
| 18 | Resynchronization after out of service or after simulation (Page 213) |
| 19 | Resynchronization after CSF (Page 214) |
| 20 | Resynchronization with StateTrkOn input (Page 214) |
| 21 | Control with output StopCtrl and state “No Cmd” (Page 214) |

### Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 239).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Output signal as a pulse signal or static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 57). In addition to the static control output `Out`, the block also has pulse outputs `P_Open`, `P_Close`, which are dependent on the static control output.

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### Button labels

This block provides the standard function Labeling of buttons and text (Page 235)

Instance-specific text can be configured for the following parameters:

- `OpenMan`
- `CloseMan`

### See also

VlvL messaging (Page 1705)

VlvL I/Os (Page 1707)

VlvL modes (Page 1694)

VlvL block diagram (Page 1715)

EventTs functions (Page 2046)

## 8.10.4 VlvL error handling

### Error handling of VlvL

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Mode switchover error
- Invalid or not plausible input commands
- Control system fault (CSF)

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. |
| 41 | The value for the `LocalSetting` I/O is not within the approved limit of 0 to 4. |
| 42 | `LocalSetting` = 0 or `LocalSetting` = 3 or `LocalSetting` = 4 and `LocalLi` = 1 |
| 51 | `AutModLi` = 1 and `ManModLi` = 1<br>`OpenLocal` = 1 and `CloseLocal` = 1<br>`OpenAut` = 1 and `CloseAut` = 1<br>`OpenForce` = 1 and `CloseForce` = 1 |
| 52 | `LocalAct` = 1 and `LocalSetting` = 2 or `4` and `SimOn` = 1 |

### Mode switchover error

This error can be output by the block. Refer to the Error handling (Page 128) section for more on this.

### Invalid or not plausible input commands

This error can be output by the block. Refer to the Error handling (Page 128) section for more on this.

For the following invalid or not plausible input commands, the control output can be kept or switched to the neutral position. This depends on the function Control priority in the event of invalid or not plausible input commands (Page 186).

| **Operating mode** | **Invalid or not plausible input commands** | **Control reaction with `Feature2.Bit3` = 1** |
|---|---|---|
| Local: `Localsetting` = 1 or `Localsetting` = 3 | Pushbutton operation for local mode (`Feature2.Bit4` = 0):<br>`OpenLocal` = 1 and `CloseLocal` = 1 | Valve is set to its neutral position. |
| Local: `Localsetting` = 1 or `Localsetting` = 3 and forcing | `OpenForce` = 1 and `CloseForce` = 1 | |
| Forcing and no "local mode" | `OpenForce` = 1 and `CloseForce` = 1 | |
| "Automatic mode" and no forcing | Pushbutton operation (`Feature2.Bit4` = 0):<br>`OpenAut` = 1 and `CloseAut` = 1 | |
| "Manual mode" and no forcing | `OpenMan` = 1 and `CloseMan` = 1 | |

### Control system fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 128) section for more on this.

### See also

Description of VlvL (Page 1691)

VlvL modes (Page 1694)

VlvL block diagram (Page 1715)

VlvL I/Os (Page 1707)

VlvL messaging (Page 1705)

VlvL functions (Page 1696)

## 8.10.5 VlvL messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Instance-specific messages

### Process control fault

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 1 | AS process control message - fault | $$BlockComment$$<br>Feedback error |
| | SIG 2 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If it changes to CSF = 1, a process control fault is triggered (MsgEvId1, SIG 2).

### Instance-specific messages

You can use up to three instance-specific messages with this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 3 | AS process control message - fault | $$BlockComment$$<br>External message 1 |
| | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External message 2 |
| | SIG 5 | AS process control message - fault | $$BlockComment$$<br>External message 3 |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance **MsgEvId1**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |
| 4 | ExtVa104 |
| 5 | ExtVa105 |
| 6 | ExtVa106 |
| 7 | ExtVa107 |
| 8 | ExtVa108 |
| 9 | Reserved |
| 10 | Reserved |

The associated values 4 ... 8 are allocated to the parameters ExtVa104 ... ExtVa108 and can be used. See the "Process Control System PCS 7 - Engineering System" manual.

### See also

VlvL modes (Page 1694)

VlvL block diagram (Page 1715)

VlvL error handling (Page 1703)

## 8.10.6 VlvL I/Os

I/Os of VlvL

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutModLi* | 1= "Automatic mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutModOp* | 1 = "Automatic mode" via operator (controlled by ModLiOp = 1) | BOOL | 0 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkTypeData* | This input can not be edit or linked and is for internal usage only! | | |
| BypProt | 1 = Bypassing interlock is active in "local mode" and in simulation | BOOL | 0 |
| CloseAut* | 1 = Select Close valve in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CloseForce | 1 = Force valve closure | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| CloseLocal | 1 = Select Close valve in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CloseMan* | 1 = Select Close valve in "manual mode" | BOOL | 0 |
| CSF | 1 = External error (control system fault) Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CtrlChnST | Signal status of output channel Ctrl<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| EN | 1 = Called block will be processed | BOOL | 1 |
| EventTsIn | For interconnecting data between a technological block and the message blocks EventTs, Event16Ts.<br>The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs block. When this interconnection is configured, the messages of the EventTs block are displayed on the OS in the alarm view of the technological block and can also be acknowledged there. | ANY | |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg3 | Binary input for freely selectable message 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa104 | Associated value 4 for messages (MsgEvID1) | ANY | |
| ExtVa105 | Associated value 5 for messages (MsgEvID1) | ANY | |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| ExtVa108 | Associated value 8 for messages (MsgEvID1) | ANY | |
| FaultExt | 1 = External error<br>Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| FbkOpen | 1 = Valve open feedback signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| FbkClose | 1 = Valve closed feedback signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| Feature | I/O for additional functions (Page 1696) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| Feature2 | I/O for additional functions (Page 1696) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| Intlock | 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared<br>1 = Interlock not activated | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| Intl_En | 1 = Interlock without reset (interlock, Intlock parameter) is active | BOOL | 1 |
| LocalLi | 1 = Activate "local mode" via plant signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalOp* | 1 = "Local mode" via operator | BOOL | 0 |
| LocalSetting | Properties for the Local mode (Page 86) | INT | 0 |
| ManModLi* | 1 = "Manual mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManModOp* | 1= "Manual mode" via: OS operator (controlled via ModLiOp = 0) | BOOL | 1 |
| ModLiOp | Switchover of operating mode between:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Monitor | 1 = Feedback monitoring | BOOL | 1 |
| MonSafePos | 1 = Go to neutral position in the event of monitoring errors | BOOL | 1 |
| MonTiDynamic* | Monitoring time after operation in [s] | REAL | 3.0 |
| MonTiStatic* | Monitoring time for feedback errors without operation in [s] | REAL | 3.0 |
| MS_RelOp* | 1= Release for maintenance via OS operator | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NoFbkClose | 1 = No feedback present for "valve closed" | BOOL | 0 |
| NoFbkOpen | 1 = No feedback present for "valve open" | BOOL | 0 |
| Occupied | 1 = In use by a batch | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpenAut* | 1 = Select Open valve in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpenForce | 1 = Force valve opening | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpenLocal | 1 = Select Open valve in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpenMan* | 1 = Select Open valve in "manual mode" | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 1696) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| Permit | 1 = Enable for opening / closing from neutral position<br>0 = Valve activation not enabled on OS | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| PermitSafePos | 1=Valve SafePos allowed | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| PermSf_En | 1=Permit SafePos enabled | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Perm_En | 1 = Activation enable (enable, Permit parameter) is active | BOOL | 1 |
| Protect | 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block<br>1 = Protective interlocking not activated | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| Prot_En | 1 = Protective interlock (protection, Protect parameter) is active | BOOL | 1 |
| PulseWidth* | Pulse width of control signal [s] | REAL | 3.0 |
| RstLi* | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | 1 = Reset via operator | BOOL | 0 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SafePos | Neutral position for valve:<br>1= Open<br>0 = Closed | BOOL | 0 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | - |
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | - |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserAna1 | Analog auxiliary value 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UA1unit | Unit of measure for analog auxiliary value 1 | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| UserAna2 | Analog auxiliary value 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UA2unit | Unit of measure for analog auxiliary value 2 | INT | 0 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |
| WarnTiAut | Prewarning of valve movement from neutral position in "automatic mode" in [s] | REAL | 0.0 |
| StateTrkOn | Initiate tracking of state | BOOL | 0 |
| WarnTiMan | Prewarning of valve movement from neutral position in "manual mode" in [s] | REAL | 0.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutAct | 1 = "Automatic mode" enabled<br>0 = "Manual mode" is enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutoST | 1 = Worst signal status of automatic commands | BYTE | 16#80 |
| Closed | 1 = Valve is closed | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Closing | 1 = Valve is closing | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl | Control output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopCtrl | Control output go to SafePos | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CurrMon | Remaining monitoring time [s] | DINT | 0 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see VlvL error handling (Page 1703) | INT | -1 |
| FaultExtOut | FaultExt as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkCloseOut | Valve closed feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkOpenOut | Valve open feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| GrpErr | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IntlockOut | Intlock as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalAct | 1 = "Local mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LockAct | 1 = Interlock (Intlock, Permit or Protect) is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManAct | 1 = "Manual mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| MonDynErr | 1 = Feedback error due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MonStaErr | 1 = Feedback error due to unexpected feedback change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message acknowledgement status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | Alarm error 1 (output ERROR of first ALARM_8P) | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Opened | 1 = Valve is open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Opening | 1 = Valve is opening | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| P_Close | 1 = Pulse signal to close valve | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Open | 1 = Pulse signal to open valve | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Rst | 1= Pulse output for reset<br>The parameter persists for one cycle after a reset. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PermitOut | Permit as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PermitSafePosOut | PermitSafePos as output | STRUCT<br>• Value: BOOL<br>ST: BYTE | -<br>• 1<br>• 16#FF |
| ProtectOut | Protect as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToReset | 1 = Ready for reset via RstLi input or commands in "local", "automatic", or "manual mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToStart | 1 = Active start readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 1691) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1691) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1691) | DWORD | 16#00000000 |
| UA1unitOut | UA1 as output | INT | 0 |
| UA2unitOut | UA2 as output | INT | 0 |
| UserAna1Out | UserAna1 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| UserAna2Out | UserAna2 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| ResyncST | Worst Signal Status of inputs for resynchronization | BYTE | 16#80 |
| WarnAct | 1 = Prewarning for valve movement away from neutral position active (parameters WarnTiAut and WarnTiMan) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### See also

VlvL messaging (Page 1705)

VlvL modes (Page 1694)

VlvL block diagram (Page 1715)

Error handling (Page 128)

## 8.10.7 VlvL block diagram

### VlvL block diagram

A block diagram is not provided for this block.

### See also

Description of VlvL (Page 1691)

VlvL modes (Page 1694)

VlvL error handling (Page 1703)

VlvL messaging (Page 1705)

VlvL I/Os (Page 1707)

VlvL functions (Page 1696)

## 8.10.8 Operator control and monitoring

### 8.10.8.1 VlvL views

#### Views of the VlvL block

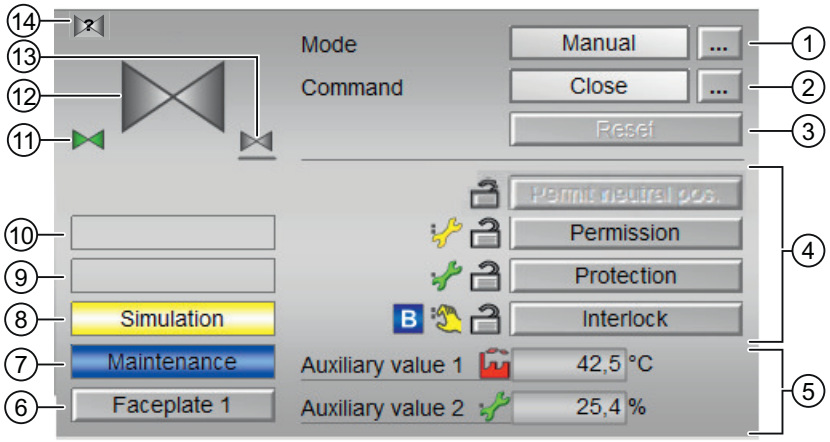
The block VlvL provides the following views:

- VlvL standard view (Page 1717)
- Alarm view (Page 336)
- Trend view (Page 341)
- Parameter view for motors and valves (Page 320)
- VlvL preview (Page 1721)
- Memo view (Page 338)
- Batch view (Page 336)
- VlvL block icon (Page 1725)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 8.10.8.2 VlvL standard view

#### VlvL standard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual and automatic mode for motors, valves and dosers (Page 82)
- Manual and automatic mode for motors, valves and dosers (Page 82)
- Local mode (Page 86)
- Out of service (Page 77)

Refer to section Switching operating states and operating modes (Page 289) for information on switching the operating mode.

#### (2) Opening and closing the valve

This area shows you the default operating state for the valve. The following states can be shown and executed here:

- "Open"
- "Close"

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

If the text is configured for these commands, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

#### (3) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in the section Resetting the block in case of interlocks or errors (Page 47).

### (4) Operating range for the interlock functions of the block

This display is only visible when the corresponding block input is connected.

You can use this button to control the interlock functions of the block. You can find additional information on this in the section Interlocking functions (Page 107).

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 113)), for example:

- Signal status (see Forming and outputting the signal status for technologic blocks (Page 117)), for example:

- Bypass information (see Forming the group status for interlock information (Page 113)):

If the text is configured for the interlocks, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (5) Display of auxiliary values

This display is only visible when the corresponding block input is connected.

You can use this area to display two auxiliary values that have been configured in the engineering system (ES). You can find additional information on this in the section Displaying auxiliary values (Page 239).

### (6) Navigation button for switching to the standard view of any faceplate

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234).

### (7) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this is available in section Release for maintenance (Page 70) Display area for block states.

### (8) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"
- "Delay"

You will find more detailed information on this in the sections Simulating signals (Page 64) and Display of delay times (Page 288).

### (9) Display area for block states

This area provides additional information on the operating state of the block:

- "External error"
- "End position error"
- "Control error"
- "Internal error"
- "Changeover error"
- "Invalid Signal"

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 104) , Error handling (Page 128) (section "Invalid input signals" and "Mode switchover error") and Motor protection function (Page 106).

### (10) Display area for block states

This area provides additional information on the operating state of the block:

- "Forced open"
- "Forced close"
- "Request 0/1": Reset is expected but reset is not possible from faceplate. For more details, refer:Reset of errors during Request 0/1 (Page 382)

You can find additional information on this in the section Forcing operating modes (Page 45).

### (11) Automatic preview

This display is only visible in "manual mode", in "local mode", or with a reset request in "automatic mode", when the current output signals are not identical to the control in "automatic mode".

The display shows what state the valve would assume if you switched from "manual" or "local" mode to "automatic mode", or performed a reset to "automatic mode".

### (12) Status display of the valve

The current status of the valve is graphically displayed here.

You can find more information about this in section VlvL block icon (Page 1725)

### (13) Neutral position of the valve

Display the neutral position for the valve:

- If the neutral position of the valve is "Closed" (`SafePos` = 0), , a gray valve is shown.
- If the neutral position of the valve is "Open" (`SafePos` = 1), a green valve is shown.

### (14) Mini preview for Resynchronization

Displays the mini preview for Resynchronizing the signal state.

The mini preview displays the following states:

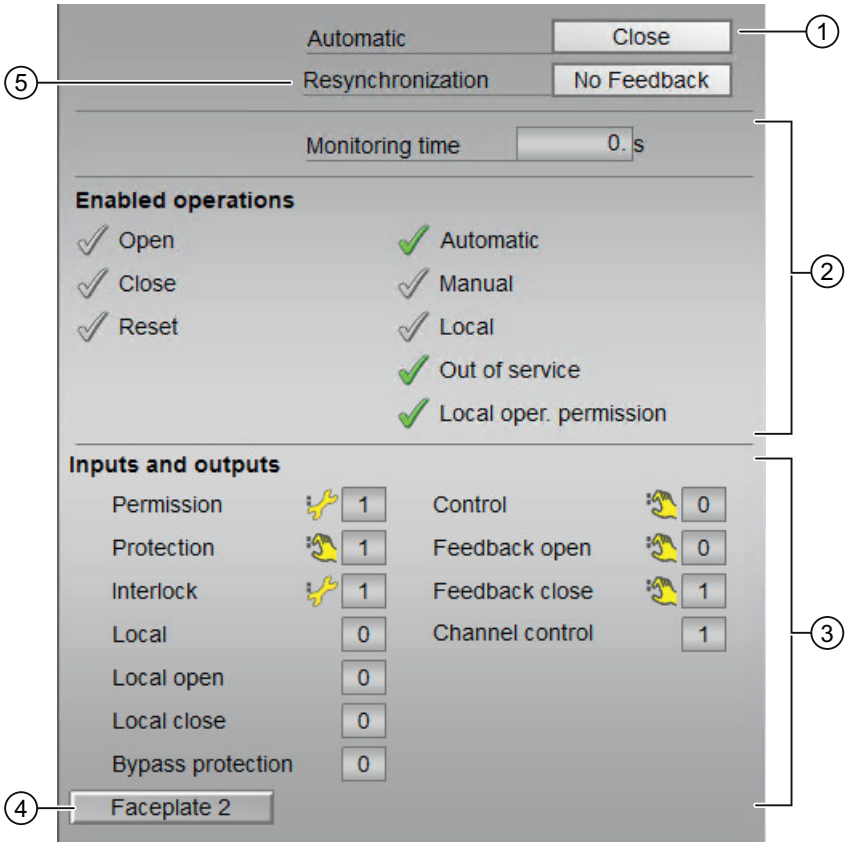
- Open
- Close
- No Cmd

**Note**

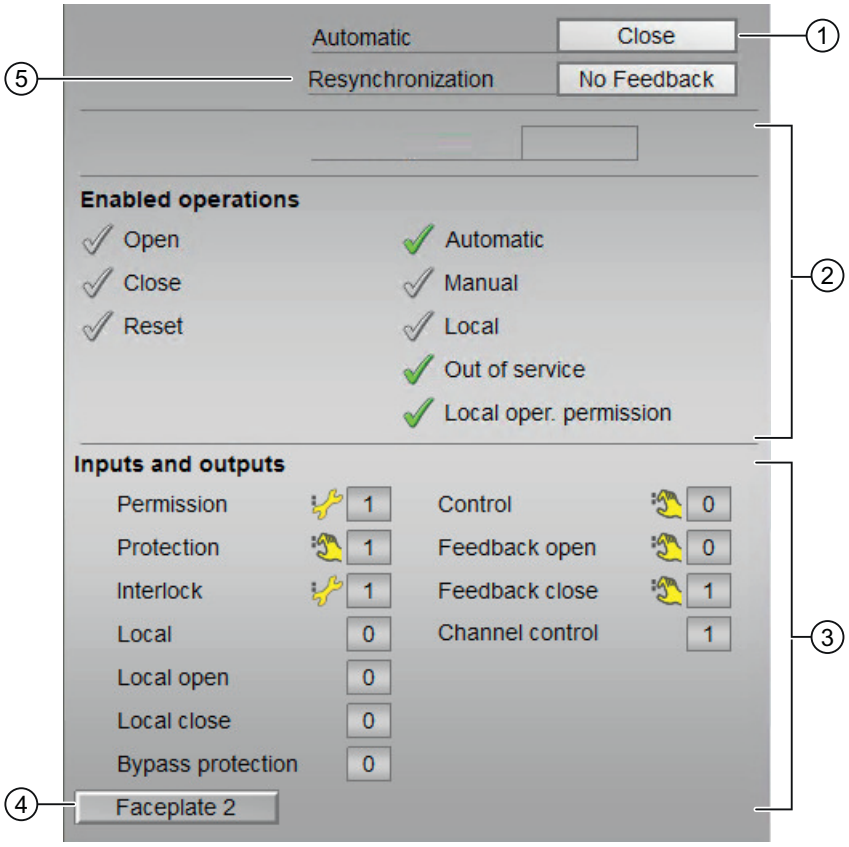
Mini preview for resync is visible only if resynchronization command is different to control output value.

### 8.10.8.3 VlvL preview

**Preview of VlvL**




Display of the current monitoring time is visible.




Display of the current monitoring time is not visible.

### (1) Automatic preview

This area shows you the block status after it has switched from the "Manual" mode or "Local" mode to the "Automatic" mode.

If the block is in "Automatic" mode, the current block state is displayed.

**Note**

- If the valve is closed and interlock locking is active (`SafePos`=0 Close), the automatic preview does not show the current status "Closed", instead it shows the active automatic command "Open".
- If the valve is in the end position and the automatic stop command is active, the automatic preview does not show the current status "Closed", but the active automatic command "Stop".

This area also displays the worst signal status of the following automatic commands:

- `OpenAut`
- `CloseAut`

### (2) Enable operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "Open": You can open the valve.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Close": You can close the valve.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Reset": You can reset the valve if interlocks or errors occur.
- "Automatic": You can switch to "automatic mode".
- "Manual": You can switch to "manual mode".
- "Local": You can switch to "local mode".
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operator permission": Use the ← button to switch to the standard view of the OpStations block. You can find additional information on this in the section Operator control permissions (Page 285).
- "Monitoring time": Display of the remaining monitoring time.

### (3) Displaying current control signals

This area shows the most important parameters for this block with the current selection:

- "Permission":
  This display is only visible when the corresponding block input is connected.
  - 0 = Valve activation not enabled on OS
  - 1 = Enable for "opening"/"closing" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Permit neutral pos" (`Feature2.Bit16` = 0):
  This display is only visible when the corresponding block input is connected.
  - 0 = prevents stop command to the motor
  - 1 = prevents the control in safe position direction

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection":
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock":
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Local": 1 = Control signal for "Local mode" (LocalLi) is active
- "Local open": 1 = Opening the valve in "local mode"
- "Local close": 1 = Closing the valve in "local mode"
- "Interlock deact.":
  - 0 = Bypass disabled
  - 1 = Bypassing interlock in "local mode" and in "simulation"
- "Control": Display for valve control:
  - 0 = Valve is closing
  - 1 = Valve is opening
- "Feedback open": 1 = Valve is open

- "Feedback close": 1 = Valve is closed
- "Channel Control": Control signal of the output channel block

### (4) Navigation button for switching to the standard view of any faceplate

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

For more information, see section Opening additional faceplates (Page 234).

### (5) Resynchronization

Displays the Resynchronization state and Resynchronization Signal status based on feedback value.

## 8.10.8.4 VlvL block icon

### Block icons for VlvL

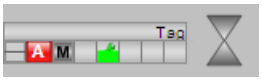
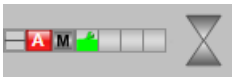
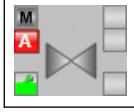
A variety of block icons are available with the following functions:

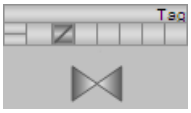
- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as control system faults
- Operating modes
- Signal status, release for maintenance
- Forcing states
- Displays for bypassing interlocks
- Interlocks
- Memo display
- Valve status display

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | Non-rotating block icon |
|  | 4 | Non-rotating block icon |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | |
|  | 6 | |
|  | 7 | |
|  | 8 | |
|  | 9 | |

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 10 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

## Valve status display

The following valve states are shown here:

| Icon | Meaning |
|---|---|
|  | Valve open |
|  | Valve closed |
|  | Error at valve |
|  | Valve is opening |
|  | Valve is closing |
|  | Valve closed |
|  | Valve is closing |
|  | "No Cmd" state Symbol |

# 8.11 VlvS - Valve (small)

## 8.11.1 Description of VlvS

### Object name (type + number) and family

Type + number: FB 1911

Family: Drives

### Area of application for VlvS

The block is used for the following applications:

- Controlling a valve in two positions ("open"/"closed") with adjustable neutral position

**Note**

This block is also available as a large block. A comparison of the VlvL and VlvS blocks is available in the section: VlvL compared to VlvS (Page 1372)

### How it works

The valve is opened or closed by a control signal. The signal 0 corresponds to the de-energized state (neutral position) of the valve.

The control is monitored by the "open"/"close" (feedback) signals. Missing feedback can be derived from the control in the block.

Various inputs are available for control purposes. The next sections provide more detailed information on configuration, operating principles, visualization and operation.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section VlvS I/Os (Page 1739).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `AutoAct.Value = 1; ManAct.Value = 0` |
| 6 | `LocalAct.Value` |
| 7 | `LockAct.Value` |
| 8 | "Open"/"Closed" command (1 = "Open") |
| 9 | `FbkOpenOut.Value` |
| 10 | `FbkCloseOut.Value` |
| 11 | Feedback error without control change |
| 12 | Feedback error due to control change |
| 13 | `BypProt` |
| 14 | Invalid signal status |
| 15 | Not used |
| 16 | 1 = `Intlock` is active |
| 17 - 21 | Not used |
| 22 | Automatic preview (1 = "Open") |
| 23 | Not used |
| 24 | `SafePos` |
| 25 - 27 | Not used |
| 28 | For the Error status display in the Closed valve |
| 29 | For the Error status display in the Opened valve |
| 30 | External error generated by FaultExt or external control system fault CSF with set `Feature.Bit18` Activating error state for external process control error CSF (Page 162) |
| 31 | 1 = Input parameter `CtrlChnST` is interconnected |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `MsgLock` |
| 1 | Not used |
| 2 | Display for interlocks in block icon |
| 3 - 15 | Not used |
| 16 | 1 = Input parameter `FbkClose` is interconnected |
| 17 | 1 = Input parameter `FbkOpen` is interconnected |
| 18 | Reset request in automatic |

| Status bit | Parameter |
|---|---|
| 19 | 1 = No impact of input signals on "local mode" when `LocalSetting` = 0, `LocalSetting` = 2 and OOS mode. |
| 20 | 1 = Valve open |
| 21 | 1 = Valve closed |
| 22 | 1 = Valve opens |
| 23 | 1 = Valve closes |
| 24 - 29 | Not used |
| 30 | Bypass information for display in block icon |
| 31 | `MS_RelOp` |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 - 7 | Not used |
| 8 | "Interlock" button is enabled |
| 9 - 11 | Not used |
| 12 | Hidden bypass signal in interlock |
| 13 | Not used |
| 14 | `Feature2.Bit2`: Separate bypass signal |
| 15 - 25 | Not used |
| 26 | Show automatic preview in the standard view |
| 27 | Not used |
| 28 | `GrpErr.Value` |
| 29 | `RdyToStart.Value` |
| 30 | Not used |
| 31 | Internal error |

### See also

VlvS modes (Page 1731)
VlvS functions (Page 1732)
VlvS error handling (Page 1736)
VlvS reporting (Page 1738)
VlvS block diagram (Page 1745)

## 8.11.2 VlvS modes

### VlvS operating modes

The block supports all standard modes:

- Local mode (Page 86)
- Automatic mode (Page 82)
- Manual mode (Page 82)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

### "Local mode"

You can find general information on "Local mode", switching modes and Bumpless switchover in the Local mode (Page 86) section.

**Note**

**"Local mode" for the block VlvS**

In contrast to the "Large" blocks, it is only possible to perform settings in this block `LocalSetting` with 0, 2 and 5.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and Bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 82) section.

Valve actions you can control in "automatic mode":

- "Open" (`OpenAut` = 1)
- "Close" (`CloseAut` = 1)

**Note**

**Information about the "Small" block**

This "Small" block works with pushbutton operation. The automatic commands are therefore latching, in other words, `OpenAut` and `CloseAut` can be reset to 0 after the control is changed. In "manual" and "local" modes, however, the automatic commands are not saved and in the absence of automatic commands the automatic control is tracked.

### "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the section Manual and automatic mode for motors, valves and dosers (Page 82).

Valve actions you can control in "manual mode":

- "Open" (`OpenMan` = 1)
- "Close" (`CloseMan` = 1)

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature2.Bit1` =0 and `Feature2.Bit1` =1 is as shown below:

| Output Parameters | **`Feature2.Bit1` =0** | **`Feature2.Bit1` =1** |
|---|---|---|
| FbkOpenOut.ST | 16#80 | 16#60 |
| FbkCloseOut.ST | 16#80 | 16#60 |
| Opened.ST | 16#80 | 16#60 |
| Closed.ST | 16#80 | 16#60 |
| Opening.ST | 16#80 | 16#60 |
| Closing.ST | 16#80 | 16#60 |

### See also

Description of VlvS (Page 1728)

VlvS functions (Page 1732)

VlvS error handling (Page 1736)

VlvS reporting (Page 1738)

VlvS I/Os (Page 1739)

VlvS block diagram (Page 1745)

## 8.11.3 VlvS functions

### Functions of VlvS

The functions for this block are listed below.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" |
| 1 | 1 = Operator can switch to "manual mode" |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can open the valve |
| 5 | 1 = Operator can close the valve |
| 6 | 1 = Operator can reset the valve |
| 7 | 1 = Operator can define the monitoring time for startup |
| 8 | Not used |
| 9 | 1 = Operator can activate the monitoring time function (Bit 7) |
| 10 | 1 = Operator can switch to Simulation |
| 11 | 1 = Operator can activate the Release for maintenance function |
| 12 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

## Button labels

This block provides the standard function Labeling of buttons and text (Page 235)

Instance-specific text can be configured for the following parameters:

- `OpenMan`
- `CloseMan`

## Interlocks

This block provides the following interlocks:

- Interlock without reset ("Interlock")

Refer to the Interlocks (Page 107) section for more on this.

## Disabling interlocks

This block provides the standard function Disabling interlocks (Page 111).

## Resetting the block in case of interlocks

This block provides the standard function Resetting the block in case of interlocks or errors (Page 47).

### Forming the group status for interlocks

This block provides the standard function Forming the group status for interlock information (Page 113).

### External error (`FaultExt`), external control system fault (`CSF`)

This block provides the possibility to pass an external error via the `FaultExt` parameter or an external control system fault via the CSF parameter. See VlvS error handling (Page 1736)

### Group error

This block provides the standard function Outputting group errors (Page 130)

The following parameters are taken into consideration when forming the group error:

- `CSF`
- `MonDynErr`
- `MonStaErr`
- `FaultExt`

### Neutral position

This block provides the standard function Neutral position for motors, valves and controllers (Page 54). The neutral position (de-energized state) is set using the `SafePos` parameter.

- `SafePos = 0` means that at `Ctrl = 0` the valve drive closes and at `Ctrl = 1` it opens (de-energized state is "closed")
- `SafePos = 1` means that at `Ctrl = 0` the valve drive opens and at `Ctrl = 1` it closes (de-energized state is "open")

### Outputting a signal for start readiness

This block provides the standard function Outputting a signal for start readiness (Page 59).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status ST_Worst for the block is formed from the following parameters:

- `FbkOpenOut.ST`
- `FbkCloseOut.ST`
- `LocalLi.ST`
- `CtrlChn.ST`
- `OpenAut.ST` (only if `Feature2.Bit10` = 1)
- `CloseAut.ST` (only if `Feature2.Bit10` = 1)

### Considering bad quality of automatic commands or external values

If the `Feature2` bit Considering bad quality of automatic commands or external values (Page 200) is set to 1 and one of the following parameters has bad signal status (16#00 or 16#28), the block goes to the neutral position in the "Automatic" mode:

- `OpenAut.ST`
- `CloseAut.ST`

### Feedback monitoring

This block provides the standard function Monitoring the feedbacks (Page 104). Startup characteristics are monitored by setting parameter`MonTiDynamic`. The parameter `MonTiStatic` monitors compliance with the position.

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

### Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 230).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 3 | Enabling resetting of commands for the control settings (Page 172) |
| 9 | Resetting via input signals in the event of interlocking (Protection) or errors (Page 174) |
| 10 | Exiting local mode (Page 189) |
| 11 | Activating the run time of feedback signals (Page 164) |
| 18 | Activating error state for external process control error CSF (Page 162) |
| 25 | Suppression of all messages (Page 185) |

| Bit | Function |
|---|---|
| 27 | Interlock display with LocalSetting 2 or 5 (Page 189) |
| 30 | Set reset depending on the operating mode or the LiOp parameter (Page 174) |
| 31 | Activating reset of protection / error in manual mode (Page 175) |

### Configurable reactions using the `Feature2` parameter

| Bit | Function |
|---|---|
| 1 | Changes signal status of outputs in OOS  (Page 208) |
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 3 | Control priority in the event of invalid or not plausible input commands (Page 186) |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) |
| 10 | Considering bad quality of automatic commands or external values (Page 200) |

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of VlvS (Page 1728)

VlvS modes (Page 1731)

VlvS reporting (Page 1738)

VlvS I/Os (Page 1739)

VlvS block diagram (Page 1745)

Disabling feedback for valves (Page 106)

Selecting a unit of measure (Page 240)

Enabling local operator authorization (Page 169)

## 8.11.4 VlvS error handling

### Error handling of VlvS

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Mode switchover error

- Invalid input signals
- Control system fault (CSF)

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. |
| 41 | The value for the `LocalSetting` I/O is not within the valid limit of 0, 2 or 5 |
| 42 | `LocalSetting` = 0 and `LocalLi` = 1 |
| 51 | `AutModLi` = 1 and `ManModLi` = 1<br>`OpenAut` = 1 and `CloseAut` = 1 |
| 52 | `LocalAct` = 1 and `LocalSetting` = 2 or 5 and `SimOn` = 1 |

### Mode switchover error

This error can be output by the block, see the section Error handling (Page 128).

### Invalid input signals

This error can be output by the block, see the section Error handling (Page 128).

For the following invalid input signals, the control output can be kept or switched to the neutral position. This depends on the function Control priority in the event of invalid or not plausible input commands (Page 186).

| Operating mode | Invalid input signals | Control reaction with `Feature2.Bit3` =1 |
|---|---|---|
| "Automatic mode" | `OpenAut` = 1 and `CloseAut` = 1 | Valve is set to its neutral position. |
| "Manual mode" | `OpenMan` = 1 and `CloseMan` = 1 | |

### Control system fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 128) section for more on this.

### See also

Description of VlvS (Page 1728)

VlvS modes (Page 1731)

VlvS functions (Page 1732)

VlvS reporting (Page 1738)

VlvS I/Os (Page 1739)
VlvS block diagram (Page 1745)

## 8.11.5 VlvS reporting

### Messaging

The following messages can be generated for this block:

- Process control fault
- Instance-specific messages

### Process control fault

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId1` | SIG 1 | AS process control message - fault | $$BlockComment$$<br>Feedback error |
| | SIG 2 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter `CSF`. If it changes to `CSF` = 1, a process control fault is triggered (`MsgEvId1`, SIG 2).

### Instance-specific messages

You have the option to use two instance-specific messages for this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId1` | SIG 3 | AS process control message - fault | $$BlockComment$$<br>External message 1 |
| | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External message 2 |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance MsgEvId1

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |
| 4 | ExtVa104 |
| 5 | ExtVa105 |
| 6 - 10 | Reserved |

The associated values 4 ... 5 are allocated to the parameters ExtVa104 ... ExtVa105 and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of VlvS (Page 1728)

VlvS modes (Page 1731)

VlvS functions (Page 1732)

VlvS error handling (Page 1736)

VlvS I/Os (Page 1739)

VlvS block diagram (Page 1745)

## 8.11.6 VlvS I/Os

I/Os of VlvS

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutModLi* | 1= "Automatic mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutModOp | 1 = "Automatic mode" via operator (controlled by ModLiOp = 1) | BOOL | 0 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BypProt | 1 = Bypassing interlock is active in "local mode" and in simulation | BOOL | 0 |
| CloseAut* | 1 = Select Close valve in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CloseMan* | 1 = Select Close valve in "manual mode" | BOOL | 0 |
| CSF | 1 = External error (control system fault) Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CtrlChnST | Signal status of output channel Ctrl<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| EN | 1 = Called block will be processed | BOOL | 1 |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa104 | Associated value 4 for messages (MsgEvID1) | ANY | |
| ExtVa105 | Associated value 5 for messages (MsgEvID1) | ANY | |
| FaultExt | 1 = External error<br>Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkOpen | 1 = Valve open feedback signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| FbkClose | 1 = Valve closed feedback signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| Feature | I/O for additional functions (Page 1732) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| Feature2 | I/O for additional functions (Page 1732) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| Intlock | 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared<br>1 = Interlock not activated | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| Intl_En | 1 = Interlock without reset (interlock, Intlock parameter) is active | BOOL | 1 |
| LocalLi | 1 = Activate "local mode" via plant signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalSetting | Properties for the Local mode (Page 86) | INT | 0 |
| ManModLi* | 1 = Manual mode via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManModOp* | 1 = Manual mode via: OS operator (controlled via ModLiOp = 0) | BOOL | 1 |
| ModLiOp | Switchover of operating mode between:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Monitor | 1 = Feedback monitoring | BOOL | 1 |
| MonSafePos | 1 = Go to neutral position in the event of monitoring errors | BOOL | 1 |
| MonTiDynamic* | Monitoring time after operation in [s] | REAL | 3.0 |
| MS_RelOp* | 1= Release for maintenance via OS operator | BOOL | 0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Occupied | 1 = In use by a batch | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpenAut* | 1 = Select Open valve in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpenMan* | 1 = Select Open valve in "manual mode" | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 1732) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| RstLi* | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | 1 = Reset via operator | BOOL | 0 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SafePos | Neutral position for valve:<br>1= Open<br>0 = Closed | BOOL | 0 |
| SimOn | 1 = Simulation on | BOOL | 0 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutAct | 1 = "Automatic mode" enabled<br>0 = "Manual mode" is enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutoST | 1 = Worst signal status of automatic commands | BYTE | 16#80 |
| Ctrl | Control output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Closed | 1 = Valve is closed | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Closing | 1 = Valve is closing | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see VlvS error handling (Page 1736) | INT | -1 |
| FaultExtOut | FaultExt at output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkCloseOut | Valve closed feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkOpenOut | Valve open feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| GrpErr | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IntlockOut | Intlock as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalAct | 1 = "Local mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| LockAct | 1 = Interlock is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManAct | 1 = "Manual mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| MonDynErr | 1 = Feedback error due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MonStaErr | 1 = Feedback error due to unexpected feedback change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Opened | 1 = Valve is open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Opening | 1 = Valve is opening | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| P_Rst | 1= Pulse output for reset<br>The parameter persists for one cycle after a reset. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToReset | 1 = Ready for reset via RstLi input or commands in "local", "automatic", or "manual mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToStart | 1 = Active start readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 1728) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1728) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1728) | DWORD | 16#00000000 |

### See also

VlvS modes (Page 1731)
VlvS block diagram (Page 1745)
VlvS reporting (Page 1738)
Error handling (Page 128)

## 8.11.7 VlvS block diagram

### VlvS block diagram

A block diagram is not provided for this block.

### See also

Description of VlvS (Page 1728)
VlvS modes (Page 1731)
VlvS functions (Page 1732)
VlvS error handling (Page 1736)
VlvS I/Os (Page 1739)
VlvS reporting (Page 1738)

## 8.11.8 Operator control and monitoring

### 8.11.8.1 VlvS views

#### Views of the VlvS block

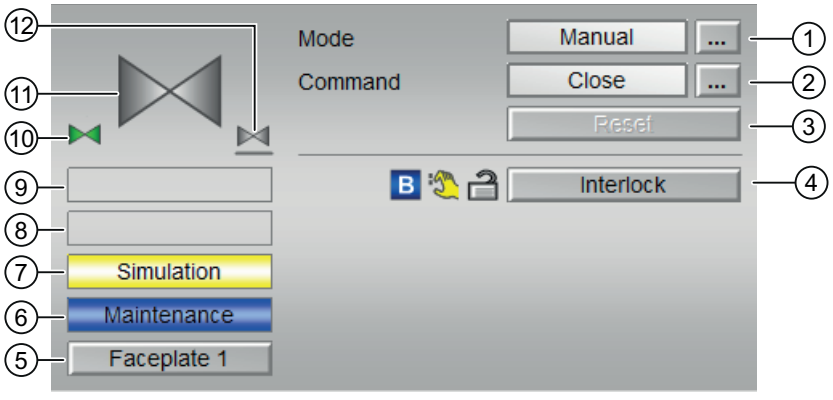
The block VlvS provides the following views:

- VlvS standard view (Page 1746)
- Alarm view (Page 336)
- Trend view (Page 341)
- Parameter view for motors and valves (Page 320)
- VlvS preview (Page 1749)
- Memo view (Page 338)
- Batch view (Page 336)
- VlvS block icon (Page 1751)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 8.11.8.2 VlvS standard view

#### VlvS standard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual and automatic mode for motors, valves and dosers (Page 82)
- Manual and automatic mode for motors, valves and dosers (Page 82)

- Local mode (Page 86)
- Out of service (Page 77)

Refer to section Switching operating states and operating modes (Page 289) for information on switching the operating mode.

### (2) Opening and closing the valve

This area shows you the default operating state for the valve. The following states can be shown and executed here:

- "Open"
- "Close"

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

If the text is configured for these commands, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (3) Resetting the block

Click "Reset" for errors. You can find additional information on this in the section Resetting the block in case of interlocks or errors (Page 47).

### (4) Operating range for the interlock functions of the block

This display is only visible when the corresponding block input is connected.

You can use this button to control the interlock functions of the block. You can find additional information on this in the section Interlocks (Page 107).

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 113)), for example:

- Signal status (see Forming and outputting the signal status for technologic blocks (Page 117)), for example:

- Bypass information (see Forming the group status for interlock information (Page 113)):

If the text is configured for the interlocks, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (5) Navigation button for switching to the standard view of any faceplate

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234).

### (6) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this is available in section Release for maintenance (Page 70) Display area for block states.

### (7) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"

You can find additional information on this in the section Simulating signals (Page 64).

### (8) Display area for block states

This area provides additional information on the operating state of the block:

- "External error"
- "End position error"
- "Control error"
- "Internal error"
- "Invalid Signal"

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 104) , Error handling (Page 128) (section "Invalid input signals" and "Mode switchover error") and Motor protection function (Page 106).

### (9) Display area for block states

This area provides additional information on the operating state of the block:

- "Request 0/1": Reset is expected but reset is not possible from faceplate. For more details, refer: Reset of errors during Request 0/1 (Page 382)

You can find additional information on this in the section Forcing operating modes (Page 45).

### (10) Automatic preview

This display is only visible in "manual mode", in "local mode", or with a reset request in "automatic mode", when the current output signals are not identical to the control in "automatic mode".

The display shows what state the valve would assume if you switched from "manual" or "local" mode to "automatic mode", or performed a reset to "automatic mode".

### (11) Status display of the valve

The current status of the valve is graphically displayed here.

- Green: Valve is open
- Gray: Valve is closed
- Red: Fault at valve

You can find more information about this in section VlvS block icon (Page 1751).

### (12) Neutral position of the valve

Display the neutral position for the valve:

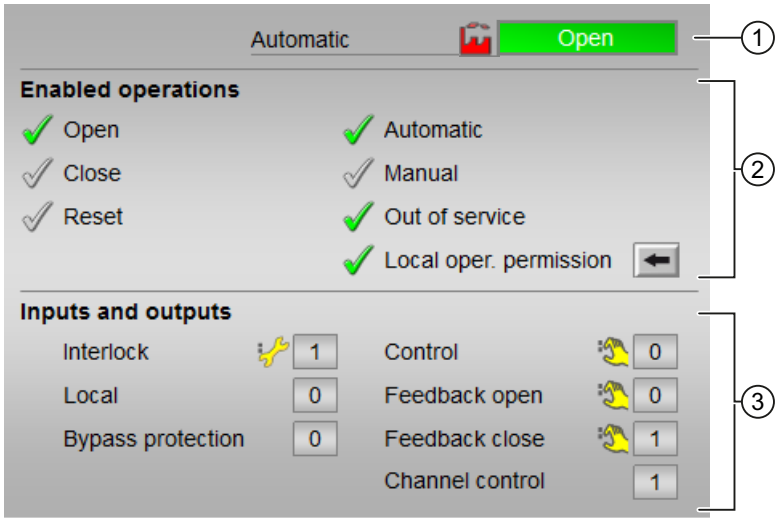
- If the neutral position of the valve is "Closed" (`SafePos` = 0), , a gray valve is shown.
- If the neutral position of the valve is "Open" (`SafePos` = 1), a green valve is shown.

### See also

Displaying auxiliary values (Page 239)

## 8.11.8.3 VlvS preview

### Preview of VlvS



### (1) Automatic preview

This area shows you the block status after it has switched from the "Manual" mode or "Local" mode to the "Automatic" mode.

If the block is in "Automatic" mode, the current block state is displayed.

**Note**

- If the valve is closed and interlock locking is active (`SafePos`=0 Close), the automatic preview does not show the current status "Closed", instead it shows the active automatic command "Open".
- If the valve is in the end position and the automatic stop command is active, the automatic preview does not show the current status "Closed", but the active automatic command "Stop".

This area also displays the worst signal status of the following automatic commands:

- `OpenAut`
- `CloseAut`

### (2) Enable operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "Open": You can open the valve.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Close": You can close the valve.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Reset": You can reset the valve if interlocks or errors occur.
- "Automatic": You can switch to "automatic mode".
- "Manual": You can switch to "manual mode".
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operator permission": Use the ← button to switch to the standard view of the OpStations block. For more information, see section Labeling of buttons and text (Page 235).

### (3) Displaying current control signals

This area shows the most important parameters for this block with the current selection:

- "Interlock":
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Local": 1 = Control signal for "Local mode" (LocalLi) is active
- "Interlock deact.":
  - 0 = Bypass disabled
  - 1 = Bypassing interlock in "local mode" and in "simulation"
- "Control": Display for valve control:
  - 0 = Valve is closing
  - 1 = Valve is opening
- "Feedback open": 1 = Valve is open
- "Feedback close": 1 = Valve is closed
- "Channel Control": Control signal of the output channel block

### See also

Operator control permissions (Page 285)

## 8.11.8.4 VlvS block icon

### Block icons for VlvS

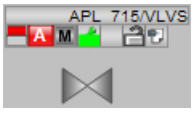
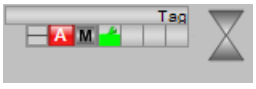
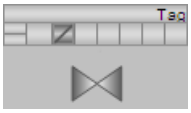
A variety of block icons are available with the following functions:

- Process tag type
- Violation of alarm, warning, and tolerance limits as well as control system faults
- Operating modes
- Signal status, release for maintenance
- Displays for bypassing interlocks
- Interlocks
- Memo display
- Valve status display

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | Non-rotating block icon |
|  | 4 | Non-rotating block icon |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | |
|  | 6 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

### Valve status display

The following valve states are shown here:

| Symbol | Meaning |
|---|---|
|  | Valve open |
|  | Valve closed |
|  | Error at valve |
|  | Valve is opening |
|  | Valve is closing |
|  | Valve closed |
|  | Valve is closing |

# 8.12 VlvMotL - Motor valve

## 8.12.1 Description of VlvMotL

### Object name (type + number) and family

Type + number: FB 1900

Family: Drives

### Area of application for VlvMotL

The block is used for the following applications:

- Motor valve control

### How it works

Various operating modes are available for controlling the motor-driven valve. This functionality allows you to set specific valve states. All changes of modes or states and faults occurring in this context are monitored, visualized in the faceplate and reported to the operator. Operators with suitable permissions can use the block icon and the faceplate to view the current states of the motor-driven valve and to operate it.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

For the VlvMotL block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 2832)
- Motor valve (ValveMotor) (Page 2867)

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

After a startup without control (`Open`, `Close` = 0)", no monitoring of the feedback signals `FbkOpen` and `FbkClose` takes place during the `V_MonTiStatic` time. Changes to `FbkOpen` and `FbkClose` are applied. This means that the feedback is monitored again, also in stop state.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section VlvMotL I/Os (Page 1778).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `AutoAct.Value` |
| 6 | `LocalAct.Value` |
| 7 | 0: Open padlock in the block icon<br>1: Closed padlock in the block icon<br>With `Feature2.Bit16` = 1, a closed padlock is displayed if "Interlock Open" or "Interlock Close" is active. |
| 8 | `Open.Value` |
| 9 | Motor is stopped |
| 10 | `Close.Value` |
| 11 | Torque shutdown enabled (`TorqOpen` or `TorqClose` = 1)<br>When the seal valve function is enabled, the torque shutoff is only active if the `TorqClose` signal is enabled before the valve feedback. |
| 12 | `WarnAct.Value` or<br>`IdleTime` active |
| 13 | Feedback error without control change |
| 14 | Feedback error due to control change |
| 15 | `Mode Switch Fail` |
| 16 | 1 = `Intlock` is active |
| 17 | 1 = `Permit` is active |
| 18 | 1 = `Protect` is active |
| 19 | `Trip.Value` |
| 20 | Display „Forced open“ |
| 21 | Display „Forced stop“ |
| 22 | Display „Forced close“ |
| 23 | "Interlock" button is enabled |
| 24 | 0 = Display neutral position "Closed" Display "Closed" for representation of neutral position in `SafePos` = 0 or 2<br>1 = Display "Opened" for representation of neutral position in `SafePos` = 1 or 2 |
| 25 | 1 = Display "Intermediate" for representation of neutral position in `SafePos` = 2 |
| 26 | Bypass information for display in block icon |
| 27 | Bypass enabled (`BypProt` = 1) and `Local.Act` = 1 or `SimOn` = 1 |
| 28 | Invalid signal status |
| 29 | 0 = closed<br>1 = open |
| 30 | "Permission" button is enabled |
| 31 | "Protection" button is enabled |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `MsgLock` |
| 1 | `AV_AH_Act.Value` |
| 2 | `AV_WH_Act.Value` |
| 3 | `AV_TH_Act.Value` |
| 4 | `AV_TL_Act.Value` |
| 5 | `AV_WL_Act.Value` |
| 6 | `AV_AL_Act.Value` |
| 7 | `AV_AH_En` OR `Feature.Bit23` |
| 8 | `AV_WH_En` OR `Feature.Bit23` |
| 9 | `AV_TH_En` OR `Feature.Bit23` |
| 10 | `AV_TL_En` OR `Feature.Bit23` |
| 11 | `AV_WL_En` OR `Feature.Bit23` |
| 12 | `AV_AL_En` OR `Feature.Bit23` |
| 13 | `AV_AH_MsgEn` |
| 14 | `AV_WH_MsgEn` |
| 15 | `AV_TH_MsgEn` |
| 16 | `AV_TL_MsgEn` |
| 17 | `AV_WL_MsgEn` |
| 18 | `AV_AL_MsgEn` |
| 19 | 1 = No impact of input signals on "local mode" with `LocalSetting` = 0, `LocalSetting` = 2, `LocalSetting` = 4 and OOS mode. |
| 20 | 1 = Valve closes |
| 21 | 1 = Valve closed |
| 22 | 1 = Valve stopped |
| 23 | 1 = Valve opens |
| 24 | 1 = Valve open |
| 25 | For the Error status display in the Closed valve |
| 26 | For the Error status display in the Opened valve |
| 27 | Automatic preview for "opening" |
| 28 | Automatic preview for "closing" |
| 29 | Automatic preview for "stopping" |
| 30 | Display for interlocks in block icon |
| 31 | `MS_RelOp` |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 | `M_MonStaErr.Value` |
| 1 | `M_MonDynErr.Value` |
| 2 | `V_MonStaErr.Value` |
| 3 | `V_MonDynErr.Value` |

| Status bit | Parameter |
|---|---|
| 4 | `M_MonStopErr.Value` |
| 5 | 1 = "Permit neutral pos." or " Permission stop" button is enabled |
| 6 | 1 = Permission SafePos / Stop is active |
| 7 | Not used |
| 8 | Reset request in automatic |
| 9 | External error generated by FaultExt or external control system fault from CSF with set `Feature.Bit18` Activating error state for external process control error CSF (Page 162) |
| 10 | Hidden bypass signal in Permission SafePos / Stop |
| 11 | Motor protection display (`Trip.Status` ≠ 16#FF) |
| 12 | 1 = Input parameter `FbkClose` is connected |
| 13 | 1 = Input parameter `FbkClosing` is connected |
| 14 | 1 = Input parameter FbkOpen is connected |
| 15 | 1 = Input parameter `FbkOpening` is connected |
| 16 | 1 = Input parameter `TorOpen` is connected |
| 17 | 1 = Input parameter `TorClose` is connected |
| 18 | `SimLiOp.Value` |
| 19 | 1 = Enable for "rapid stop"(`Feature` Bit Enabling rapid stop via faceplate (Page 179)) |
| 20 | 1 = Input parameter `OpenChnST` is interconnected |
| 21 | 1 = Input parameter `CloseChnST` is interconnected |
| 22 | Not used |
| 23 | Command for "rapid stop" |
| 24 | "Open"/"Stop" command |
| 25 | "Close"/"Stop" command |
| 26 | Show automatic preview in the standard view |
| 27 | Internal error |
| 28 | `GrpErr.Value` |
| 29 | `RdyToStart.Value` |
| 30 | Auxiliary value 1 visible |
| 31 | Auxiliary value 2 visible |

### Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 | Effective signal 1 of the message block connected via `EventTsIn` |
| 1 | Effective signal 2 of the message block connected via `EventTsIn` |
| 2 | Effective signal 3 of the message block connected via `EventTsIn` |
| 3 | Effective signal 4 of the message block connected via `EventTsIn` |
| 4 | Effective signal 5 of the message block connected via `EventTsIn` |
| 5 | Effective signal 6 of the message block connected via `EventTsIn` |
| 6 | Effective signal 7 of the message block connected via `EventTsIn` |
| 7 | Effective signal `8` of the message block connected via `EventTsIn` |
| 8 | `AV` not connected |

| Status bit | Parameter |
|---|---|
| 9 | Delay of the `AV_AH_Lim` message |
| 10 | Delay of the `AV_WH_Lim` message |
| 11 | Delay of the `AV_TH_Lim` message |
| 12 | Delay of the `AV_TL_Lim` message |
| 13 | Delay of the `AV_WL_Lim` message |
| 14 | Delay of the `AV_AL_Lim` message |
| 15 | Collection of message delays |
| 16 | "Permission Open" button is enabled |
| 17 | "Permission Close" button is enabled |
| 18 | "Interlock Open" button is enabled |
| 19 | "Interlock Close" button is enabled |
| 20 | "Protection Open" button is enabled |
| 21 | "Protection Close" button is enabled |
| 22 | 1 = "Permission Close" is active |
| 23 | Hidden bypass signal in Permit |
| 24 | Hidden bypass signal in interlock |
| 25 | Hidden bypass signal in Protect |
| 26 | `Feature2.Bit2`: Separate bypass signal |
| 27 | 1 = "Interlock Close" is active |
| 28 | 1 = "Protection Close" is active |
| 29 | Current motor monitoring time is visible |
| 30 | Current valve monitoring time is visible |
| 31 | Separate monitoring of shutdown of the motor (`Feature.Bit13`) |

### Status word allocation for `Status5` parameter

| Status bit | Parameter |
|---|---|
| 0 - 7 | Effective signal 9..16 of the message block connected via `EventTsIn` |
| 8 | 1 = Valve still closes (`Status2.Bit20`) but the motor is not running |
| 9 | 1 = Valve still opens (`Status2.Bit23`) but the motor is not running |
| 10 | Hidden bypass signal in "Permission Close" |
| 11 | Hidden bypass signal in "Interlock Close" |
| 12 | Hidden bypass signal in "Protection Close" |
| 13 | `Feature2.Bit16` Separate interlock for each direction, position, or speed (Page 180) |
| 14 | 1= Suppressed messages |
| 15 | Not used |
| 16 - 31 | Effective signal 1...16 of the message block connected via `EventTs2In` |

### Status word allocation for `Status6` parameter

| Status bit | Parameter |
|---|---|
| 0 | Resynchronization: Valve Open |
| 1 | Resynchronization: Valve Close |
| 2 | Resynchronization: Valve Stop |
| 3 | Resynchronization: Motor Opening |
| 4 | Resynchronization: Motor Closing |
| 5 | Resynchronization: Motor Stopped |
| 6 | Resynchronization: Display preview |
| 7 | Resynchronization: Display small preview symbol in standard view |
| 8 | Resynchronization preview view for Valve/Motor open |
| 9 | Resynchronization preview view for Valve/Motor close |
| 10 | Resynchronization preview view for Valve/Motor Stop |
| 11 - 31 | Not used |

### See also

VlvMotL functions (Page 1762)
VlvMotL messaging (Page 1777)
VlvMotL block diagram (Page 1791)
VlvMotL error handling (Page 1774)
VlvMotL modes (Page 1759)

## 8.12.2 VlvMotL modes

### VlvMotL operating modes

The block supports all standard modes:

- Local mode (Page 86)
- Automatic mode (Page 82)
- Manual mode (Page 82)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

### "Local mode"

You can find general information on "Local mode", switching modes and Bumpless switchover in the Local mode (Page 86) section.

Motor valve actions you can control in "local mode"

- "Open" (`OpenLocal` = 1)
- "Close" (`CloseLocal` = 1)
- "Stop" (`StopLocal` = 1).

A block operated in "local mode" is controlled either by "local" signals or by feedback signals (input parameters `FbkOpen` and `FbkClose`; if no position can be assigned, the last valid position is accepted). Configuration takes place via the input parameter `LocalSetting`.

`LocalSetting` =2,4: The motor controls and the valve position are tracked to the feedbacks.

With `Feature Bit12` =0 (motor feedback available)

```
Open = FbkOpening
```

```
Close = FbkClosing
```

```
StopCtrl = NOT FbkOpening AND NOT FbkClosing
```

When both feedbacks are switched off, the motor controls will be tracked to stop.

When both feedbacks are active (not plausible), the motor controls will be hold and the static monitoring counter starts.

When valve is in end position the motor feedbacks are monitored to stop.

The valve position is tracked to the valve feedbacks.

Valve position open = `FbkOpen`

Valve position close = `FbkClose`

Valve position intermediate = `NOT FbkClose AND NOT FbkOpen`

The valve feedbacks are monitored to not plausibility (`FbkOpen AND FbkClose`) and of reaching the end position after starting the motor.

When a valve feedback is not available (`NoFbkOpen, NoFbkClose`) the reaching of end position is adapted by a time delay of V_MonTiDynamic.

With `Feature Bit12` =1 (no motor feedback available)

The motor controls are always switched to stop.

```
Open = false
```

```
Close = false
```

```
StopCtrl = true
```

The valve position is tracked to the valve feedbacks.

Valve position open = `FbkOpen`

Valve position close = `FbkClose`

Valve position intermediate = `NOT FbkClose AND NOT FbkOpen`

The valve feedbacks are monitored to not plausibility (`FbkOpen AND FbkClose`).

If only one feedback is available it is always plausible.

| NoFbkOpen =0 | NoFbkClose =0 |
|---|---|
| Valve position open = FbkOpen<br>Valve position close = NOT FbkOpen<br>If FbkOpen becomes inactive after a delay of V_MonTiDynamic the valve position is closed. | Valve position open = NOT FbkClose<br>Valve position close = FbkClose If FbkClose becomes inactive after a delay of V_MonTiDynamic the valve position is opend. |

It is not possible to switch in intermediate position.

When all feedbacks are not available the valve position will be hold.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and Bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 82) section.

Motor valve actions you can control in "automatic mode":

- "Open" (OpenAut = 1)
- "Close" (CloseAut = 1)
- "Stop" (StopAut = 1)

### "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the section Manual and automatic mode for motors, valves and dosers (Page 82).

Motor valve actions you can control in "manual mode":

- "Open" (OpenMan = 1)
- "Close" (CloseMan = 1)
- "Stop" (StopMan = 1)

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with Feature2.Bit1 =0 and Feature2.Bit1 =1 is as shown below:

| Output Parameters | Feature2.Bit1 =0 | Feature2.Bit1 =1 |
|---|---|---|
| FbkOpenOut.ST | 16#80 | 16#60 |
| FbkCloseOut.ST | 16#80 | 16#60 |
| FbkOpngOut.ST | 16#80 | 16#60 |
| FbkClsgOut.ST | 16#80 | 16#60 |
| Opened.ST | 16#80 | 16#60 |
| Closed.ST | 16#80 | 16#60 |

| Output Parameters | `Feature2.Bit1` =0 | `Feature2.Bit1` =1 |
|---|---|---|
| Opening.ST | 16#80 | 16#60 |
| Closing.ST | 16#80 | 16#60 |
| Stop.ST | 16#80 | 16#60 |
| AV_Out.ST | 16#80 | 16#60 |

**Note**

In "Out of service" mode feedbacks signal status will be set to simulation after moving the valve to the safe position when `Feature2.bit1=1`.

**See also**

VlvMotL block diagram (Page 1791)
VlvMotL I/Os (Page 1778)
VlvMotL error handling (Page 1774)
VlvMotL functions (Page 1762)
VlvMotL messaging (Page 1777)
Description of VlvMotL (Page 1754)

## 8.12.3 VlvMotL functions

### Functions of VlvMotL

The functions for this block are listed below.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" |
| 1 | 1 = Operator can switch to "manual mode" |
| 2 | 1 = Operator can switch to "local mode" |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can stop the motor |
| 5 | 1 = Operator can open the valve |

| Bit | Function |
|---|---|
| 6 | 1 = Operator can close the valve |
| 7 | 1 = Operator can reset the valve |
| 8 | 1 = Operator can define the monitoring time for the valve startup |
| 9 | 1 = Operator can define the monitoring time for the end position of the valve |
| 10 | 1 = Operator can enable the monitoring time function of the valve (Bit 8 - 9) |
| 11 | 1 = Operator can define or change the monitoring time for startup |
| 12 | 1 = Operator can define the monitoring time for the motor status |
| 13 | 1 = Operator can enable the monitoring time function of the motor (Bit 8 - 9) |
| 14 | 1 = Operator can activate the Simulation function |
| 15 | 1 = Operator can activate the Release for maintenance function |
| 16 | 1 = Operator can change the limit (`AV`) for high alarm |
| 17 | 1 = Operator can change the limit (`AV`) for high warning |
| 18 | 1 = Operator can change the limit (`AV`) for high tolerance |
| 19 | 1 = Operator can change the limit (`AV`) for hysteresis |
| 20 | 1 = Operator can change the limit (`AV`) for low alarm |
| 21 | 1 = Operator can change the limit (`AV`) for low warning |
| 22 | 1 = Operator can change the limit (`AV`) for low tolerance |
| 23 | 1 = Operator can activate / deactivate messages via `AV_AH_MsgEn` |
| 24 | 1 = Operator can activate / deactivate messages via `AV_WH_MsgEn` |
| 25 | 1 = Operator can activate / deactivate messages via `AV_TH_MsgEn` |
| 26 | 1 = Operator can activate / deactivate messages via `AV_TL_MsgEn` |
| 27 | 1 = Operator can activate / deactivate messages via `AV_WL_MsgEn` |
| 28 | 1 = Operator can activate / deactivate messages via `AV_AL_MsgEn` |
| 29 | 1 = Operator can change the simulation value `SimAV` |
| 30 | 1 = Operator can define the monitoring time for stopping |
| 31 | 1 = Operator can execute the rapid stop |

The block has the following permissions for the `OS1Perm` parameter:

| Bit | Function |
|---|---|
| 0 - 15 | reserved |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Restart lock after changing direction of rotation or switching off the motor

Use the input parameter `IdleTime` to enter a restart lock for changing the direction of rotation or restarting the motor. When the "Stop" command is given, the motor goes immediately into "Stop" mode, and `IdleTime` starts after the feedback (`FbkOpening` and `FbkClosing` = 0) is given. The motor cannot be started again (open or close) until the `IdleTime` has expired.

### Limit monitoring of an additional analog value

This block provides the standard function Limit monitoring of an additional analog value (Page 98).

### Limit monitoring with hysteresis

This block provides the standard function Limit monitoring with hysteresis (Page 104). It is performed via the input parameter `AV_Hyst`.

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Interlocks

This block provides the following interlocks:

- `Feature2.Bit16` = 0
  - Activation enable ("Permission")
  - Interlock without reset ("Interlock")
  - Interlock with reset ("Protection")
  - Permission for safe position (“Permit neutral pos.” or “Permission stop”)
- `Feature2.Bit16` = 1
  - Activation enable Open ("Permission Open")
  - Activation enable Close ("Permission Close")
  - Interlock open without reset ("Interlock Open")
  - Interlock close without reset ("Interlock Close")
  - Interlock open with reset ("Protection Open")
  - Interlock close with reset ("Protection Close")

Refer to the section Interlocks (Page 107) as well as Influence of the signal status on the interlock (Page 112).

### Motor protection function

This block provides the standard function Motor protection function (Page 106).

### Rapid stop for motors

This block provides the standard function Rapid stop for motors (Page 115).

### Torque monitoring

The block provides torque monitoring.

The signals of the torque monitoring switches are interconnected to input parameters `TorqOpen` and `TorqClose` for opening and closing the motor valve.

An active torque shutdown is displayed in the parameter either by the value 0 or by the signal status 16#00 or 16#28.

If the torque shutdown is active, the motor is stopped. You have the option of moving the valve in the opposite direction.

If, for example, the torque shutdown is active when the valve opens, you can still close the valve.

Active torque shutoff appears in the standard view of the display area for block states.

When the "Seal valve" function is enabled via `Feature.Bit8`, the torque shutoff for the closing `TorqClose` is also evaluated (see section Sealing the valve).

### Reset of motor control after valve end position has been reached

After a motor control in the direction of one of the end positions, open or closed, the valve is opened or closed accordingly. After the end position is reached, the block is notified of this via the `FbkOpen` or `FbkClose` input. The `Feature` bit Motor stop in end position depends only on the corresponding feedback signal (Page 199) can affect the dependencies of the feedback signals to downscale the motor in the end position.

**Note**

If you deactivate one of the valve feedback signals via the inputs `NoFbkOpen, NoFbkClose,` the corresponding valve feedback signal `FbkOpen`, `FbkClose` for the motor switch off in the end position is no longer evaluated. After the valve monitoring time `V_MonTiDynamic` has expired, the corresponding output of the valve feedback signal `FbkOpenOut`, `FbkCloseOut` is set and the associated motor control `Open`, `Close` is reset again. The motor feedback signals are also monitored when the valve feedback signals are switched off.

In the case of missing valve feedback signals, the corresponding signals of the torque monitoring with the inputs `TorqOpen` or `TorqClose` must be used.

### Sealing the valve

The function is activated via `Feature.Bit8` Sealing the valve (Page 187). The seal valve function combines the query of the end position CLOSED via the input parameter `FbkClose` with the limit violation of the configured torque via the input parameter `TorqClose`. This ensures that the valve is completely sealed.

The valve is only considered completely sealed when the feedback for the end position CLOSE has been received (0->1) and the torque cutoff for Closed is enabled. The torque shutoff should

not come before the feedback in this case. The `FbkCloseOut` output shows whether the valve is sealed:

`FbkCloseOut := FbkClose.Value has been received (0->1)`

And "Torque shutoff closed is enabled"

And "Torque shutoff closed is enabled" did not come before `FbkClose` (0->1)

"Torque shutoff closed is enabled" means that `TorqClose = 0` or the signal status is `16#00` or `16#28`.

When "Torque shutoff closed is enabled" comes before the end position feedback closed, this is displayed in the faceplate in the standard view and the motor stops. Opening the valve is still possible.

**Note**

The command text for Close in the standard view and preview can be changed to Seal in the CFC at the `CloseMan` parameter in Text 1. See Section Labeling of buttons and text (Page 235)

### Disabling interlocks

This block provides the standard function Disabling interlocks (Page 111).

### Resetting the block in case of interlocks

This block provides the standard function Resetting the block in case of interlocks or errors (Page 47).

### External error (`FaultExt`), external control system fault (`CSF`)

This block provides the possibility to pass an external error via the `FaultExt` parameter or an external control system fault via the CSF parameter. See VlvMotL error handling (Page 1774)

### Group error

This block provides the standard function Outputting group errors (Page 130).

The following parameters are taken into consideration when forming the group error:

- `CSF`
- `Trip`
- `V_MonDynErr`
- `V_MonStaErr`
- `M_MonDynErr`
- `M_MonStaErr`
- `FaultExt`

### Outputting a signal for start readiness

This block provides the standard function Outputting a signal for start readiness (Page 59).

### Forming the group status for interlocks

This block provides the standard function Forming the group status for interlock information (Page 113).

### Control loop is open (CascaCut)

The output CascaCut indicates whether the control loop within this block is open. In the open control loop, the master controller must be set to tracking mode to avoid until windup behavior. If you use this block in a cascade structure, then connect the output CascaCut to the input MV_TrkOn of the master controller.

CascaCut is active in the following cases:

- Block is not in automatic mode
- Force function is active
- Motor protection is active
- Interlock is active
- Monitoring error is active

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status for the block is formed from the following parameters:

- FbkClsgOut.ST
- FbkOpngOut.ST
- FbkOpenOut.ST
- FbkCloseOut.ST
- LocalLi.ST
- OpenLocal.ST
- StopLocal.ST
- TorqClose.ST
- CloseLocal.ST
- Trip.ST
- TorqOpen.ST
- AV_Out.ST
- OpenChn.ST
- CloseChn.ST

- `OpenAut.ST` (only if `Feature2.Bit10` = 1)
- `CloseAut.ST` (only if `Feature2.Bit10` = 1)
- `StopAut.ST` (only if `Feature2.Bit10` = 1)

### Considering bad quality of automatic commands or external values

If the `Feature2` bit Considering bad quality of automatic commands or external values (Page 200) is set to 1 and one of the following parameters has bad signal status (16#00 or 16#28), the block goes to the neutral position in the "Automatic" mode:

- `OpenAut.ST`
- `CloseAut.ST`
- `StopAut.ST`

### Forcing operating modes

This block provides the standard function Forcing operating modes (Page 45).
Inputs `OpenForce` and `CloseForce` and `StopForce`force the block to open, close or stop.

### Feedback monitoring

This block provides the standard function Monitoring the feedbacks (Page 104).

The `FbkOpen` and `FbkClose` feedback is monitored for the valve; the `FbKOpening` and `FbkClosing` feedback is monitored for the motor.

The monitoring of the feedback for the valve will not be active if it was stopped during opening or closing.

#### Monitoring valve feedback

The monitoring of valve feedback is set using the parameter `V_Monitor`.

Startup characteristics are monitored by setting parameter `V_MonTiDynamic`. The parameter `V_MonTiStatic` monitors compliance with the position.

Feedback errors are displayed at the corresponding parameters `V_MonDynErr` and/or `V_MonStaErr`.

**Note**

After the motor valve stops in the intermediate position or end position or after a startup without control (`Open`, `Close` = 0)" or a stop after a monitoring error, no monitoring of the feedback signals `FbkOpen` and `FbkClose` takes place during the `V_MonTiStatic` time. Changes to `FbkOpen` and `FbkClose` are applied. This means that the feedback is monitored again, also in stop state.

**Note**

When the "Seal valve" function is enabled via `Feature.Bit8`, the torque shutoff for closing `TorqClose` is also evaluated (see section Sealing the valve (Page 187)).

**Monitoring the motor feedback**

The monitoring of motor feedback is set using the parameter M_Monitor.

Startup characteristics are monitored by setting parameterM_MonTiDynamic. The parameter M_MonTiStatic monitors compliance with the position.

Feedback errors are displayed at the corresponding parameters M_MonDynErr and/or M_MonStaErr.

## Disabling feedback

This block provides the standard function Disabling feedback for valves (Page 106). Feedback monitoring can be deactivated separately for each feedback with NoFbkOpen or NoFbkClose as required.

## Tracking the valve position after a motor stop or after a startup

Because of vibration or mass inertia after a motor stop the block tracks the valve position for the time V_MonTiStatic. During this time the static feedback monitoring is not active. The monitoring starts after the time V_MonTiStatic.

- Motor is stopped in end position open (FbkOpen =1, FbkClose =0):
  Open and intermediate position will be tracked.
  FbkOpen =1, FbkClose =0: valve goes to open position
  FbkOpen =0, FbkClose =0: valve goes to intermediate position
- Motor is stopped in intermediate position (FbkOpen =0, FbkClose =0):
  Open, Close and intermediate position can be tracked.
  FbkOpen =1, FbkClose =0: valve goes to open position
  FbkOpen =0, FbkClose =0: valve goes to intermediate position
  FbkOpen =0, FbkClose =1: valve goes to close position
- Motor is stopped in end position close (FbkOpen =0, FbkClose =1):
  Close and intermediate position can be tracked.
  FbkOpen =1, FbkClose =0: valve goes to open position
  FbkOpen =0, FbkClose =0: valve goes to intermediate position

Because of preparing the feedback signals after startup the block tracks the valve position for the time V_MonTiStatic.

- Open, Close and intermediate position can be tracked.
  FbkOpen =1, FbkClose =0: valve goes to open position
  FbkOpen =0, FbkClose =0: valve goes to intermediate position
  FbkOpen =0, FbkClose =1: valve goes to close position

**Note**

After resetting the block, in the case of interlocks or errors, the valve position will always be tracked, depending on FbkOpen and FbkClose.

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Specify warning times for control functions

This block provides the standard function Specifying warning times for control functions at motors and valves (Page 56).

You can generate warning signals when, for example, valves open. Warning signals can be generated in the following modes:

- Manual and automatic mode for motors, valves and dosers (Page 82) (Input parameter `WarnTiMan`)
- Manual and automatic mode for motors, valves and dosers (Page 82) (Input parameter `WarnTiAut`)

You specify the warning times in seconds using the input parameters `WarnTiMan` and `WarnTiAut`. If, for example, a valve opens, this is displayed at the output parameter with `WarnAct` = 1. The valve then opens after the set warning time has expired and `WarnAct` then returns to 0.

A corresponding warning is not output if the warning times (`WarnTiMan` or `WarnTiAut`) are specified with a smaller value than the `SampleTime` parameter.

**Note**

The warning is activated for each actuation that causes the motor to start, even if this means that the valve is moved to the neutral position

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Additional value (`SimAV`, `SimAV_Li`)

In the case of internal simulation with immediate tracking of feedback, it is possible to simulate a position between the open and closed state (`FbkOpenOut` = `FbkCloseOut` = 0) by means of a stop command.

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

### Neutral position

This block provides the standard function Neutral position for motors, valves and controllers (Page 54).

### Output signal as a pulse signal or static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 57). In addition to the static control outputs `Open` and `Close`, the block also has pulse outputs `P_Open`, `P_Close`, and `P_Stop`, which are dependent on the static control output.

### Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 230).

### Disabling of limits

The block provides the standard function Limits disabled in faceplate (Page 206).

### Resynchronization

This block provides the standard function Resynchronization (Page 258).

Resynchronization is switched on over feature bits:

| | |
|---|---|
| Feature2 Bit17 =1 | Resynchronization after CPU startup |
| Feature2 Bit18 =1 | Resynchronization after Simulation or after Out of service |
| Feature2 Bit19 =1 | Resynchronization after control system fault |
| Feature2 Bit20 =1 | Resynchronization with 0->1 trigger at StateTrkOn input |

If there are no restrictions the controls are tracked as follow:

**Valve**

**NoFbkOpen =0, NoFbkClose =0:**

Plausible feedback positions are tracked as control position. In case of not plausible feedback positions the controls are not changed.

**NoFbkOpen =0, NoFbkClose =1:**

If FbkOpen is active, position open is controlled. Else position close is controlled. Stop position is not possible.

**NoFbkOpen =1, NoFbkClose = 0:**

If FbkClose is active, position close is controlled. Else position open is controlled. Stop position is not possible.

**NoFbkOpen =1, NoFbkClose =1:**

Control position is not changed.

**Motor**

**Motor feedbacks are present** Feature Bit 12 =0:

Open := FbkOpening AND NOT FbkClosing AND valve position is not open;

Close := FbkClosing AND NOT FbkOpening AND valve position is not close;

StopCtrl := NOT FbkOpening AND NOT FbkClosing;

In case if there are no plausible motor feedbacks (FbkOpening =1, FbkClosing =1) the motor goes to stop control.

**Motor feedbacks are not present** Feature Bit 12 =1:

The motor goes to stop control:

Open := false;

Close := false;

StopCtrl := true;

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 3 | Enabling resetting of commands for the control settings (Page 172) |
| 4 | Setting switch or button mode (Page 177) |
| 8 | Sealing the valve (Page 187) |
| 9 | Resetting via input signals in the event of interlocking (Protection) or errors (Page 174) |
| 10 | Exiting local mode (Page 189) |
| 11 | Activating the run time of feedback signals (Page 164) |
| 12 | Motor feedback is not available (Page 168) |
| 13 | Separate monitoring time for stopping the motor (Page 180) |
| 14 | Enabling rapid stop via faceplate (Page 179) |
| 15 | Motor stop in end position depends only on the corresponding feedback signal (Page 199) |
| 17 | Enabling bumpless switchover to automatic mode for valves, motors, and dosers (Page 184) |
| 18 | Activating error state for external process control error CSF (Page 162) |
| 19 | Reset even with locked state (Page 176) |
| 20 | Disable calculation of impulse control in LocalSetting 2 and 4 (Page 200) |
| 21 | Enable bumpless switchover to "Automatic" mode for operator only (Page 183) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 23 | Limits disabled in faceplate (Page 206) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 27 | Interlock display with LocalSetting 2 or 4 (Page 189) |
| 28 | Disabling operating points (Page 156) |
| 29 | Signaling limit violation (Page 181) |
| 30 | Set reset depending on the operating mode or the LiOp parameter (Page 174) |
| 31 | Activating reset of protection / error in manual mode (Page 175) |

In switching mode (Bit 4 = 1), control is selected with the static signals `OpenAut` and `CloseAut`. If the `OpenAut` and `CloseAut` inputs are not set, the motor is stopped. Control via `StopAut` is not needed. If the "Activate command reset for control" function (Bit 3 = 1) is activated, the inputs `OpenAut` and `CloseAut` are reset to 0 after evaluation in the block.

### Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 239).

### Configurable reactions using the `Feature2` parameter

| Bit | Function |
|---|---|
| 1 | Changes signal status of outputs in OOS (Page 208) |
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 3 | Control priority in the event of invalid or not plausible input commands (Page 186) |
| 4 | Setting switch or button mode for local commands (Page 195) |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) |
| 8 | Forcing commands in the Local mode (Page 198) |
| 9 | Button mode with triggered local control commands (Page 209) |
| 10 | Considering bad quality of automatic commands or external values (Page 200) |
| 11 | Suppress MsgLock and “Out of service” mode for a connected message block (Page 202) |
| 13 | Resetting the Pending Error in Local Mode 1 and 3 (Page 210) |
| 16 | Separate interlock for each direction, position, or speed (Page 180) |
| 17 | Resynchronization after startup (Page 213) |
| 18 | Resynchronization after out of service or after simulation (Page 213) |
| 19 | Resynchronization after CSF (Page 214) |
| 20 | Resynchronization with StateTrkOn input (Page 214) |

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Button labels

This block provides the standard function Labeling of buttons and text (Page 235).

Instance-specific text can be configured for the following parameters:

- `OpenMan`
- `CloseMan`

- StopMan
- RapidStp

### Outputs for status of drive

The block has the following outputs to indicate the status of the drives. In the CFC plan, this information can be further interconnected:

| | |
|---|---|
| Closed =1 | Motor is stopped:<br>Open, Close, FbkClsgOut, FbkOpngOut =0<br>Valve is in position closed: FbkCloseOut =1 |
| Opened =1 | Motor is stopped:<br>Open, Close, FbkClsgOut, FbkOpngOut =0<br>Valve is in position opened: FbkOpenOut =1 |
| Closing =1 | Motor drives in direction of position closed:<br>Close, FbkClsgOut =1 |
| Opening =1 | Motor drives in direction of position opened:<br>Open, FbkOpngOut =1 |
| Stop =1 | Motor is stopped and position is reached:<br>Open, Close, FbkClsgOut, FbkOpngOut =0 |

**Note**

The outputs for status of drive are only can be active if there is no monitoring error:

MonDynErr =0, MonStaErr =0

### See also

Description of VlvMotL (Page 1754)
VlvMotL messaging (Page 1777)
VlvMotL I/Os (Page 1778)
VlvMotL block diagram (Page 1791)
VlvMotL modes (Page 1759)
EventTs functions (Page 2046)

## 8.12.4 VlvMotL error handling

### Error handling of VlvMotL

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Mode switchover error
- Invalid or not plausible input commands
- Control system fault (CSF)

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. |
| 41 | The value for the `LocalSetting` I/O is not within the approved limit of 0 to 4. |
| 42 | `LocalSetting` = 0 or `LocalSetting` = 3 or `LocalSetting` = 4 and `LocalLi` = 1 |
| 51 | `OpenLocal` = 1 and `StopLocal` = 1<br>`CloseLocal` = 1 and `StopLocal` = 1<br>`OpenLocal` = 1 and `CloseLocal` = 1<br>`OpenAut` = 1 and `StopAut` = 1<br>`CloseAut` = 1 and `StopAut` = 1<br>`OpenAut` = 1 and `CloseAut` = 1<br>`AutModLi` = 1 and `ManModLi` = 1<br>`OpenForce` = 1 and `StopForce` = 1<br>`CloseForce` = 1 and `StopForce` = 1<br>`OpenForce` = 1 and `CloseForce` = 1 |
| 52 | `LocalAct` = 1 and `LocalSetting` = 2 or 4 and `SimOn` = 1 |

### Mode switchover error

This error can be output by the block, see the section Error handling (Page 128).

### Invalid or not plausible input commands

This error can be output by the block, see the section Error handling (Page 128).

For the following invalid or not plausible input commands, the control output can be kept or switched to the neutral position. This depends on the function Control priority in the event of invalid or not plausible input commands (Page 186).

| **Operating mode** | **Invalid or not plausible input commands** | **Control reaction with `Feature2.Bit3` = 1** |
|---|---|---|
| Local: `Localsetting` = 1 or `Localsetting` = 3 | Pushbutton operation for local mode (`Feature2.Bit4` = 0):<br>`OpenLocal` = 1 and `CloseLocal` = 1 or<br>`OpenLocal` = 1 and `StopLocal` = 1 or<br>`StopLocal` = 1 and `CloseLocal` = 1<br>Switching mode (`Feature2.Bit4` = 1):<br>`OpenLocal` = 1 and `CloseLocal` = 1 | Motor is started in valve neutral position direction. |
| Local: `Localsetting` = 1 or `Localsetting` = 3 and forcing | `OpenForce` = 1 and `CloseForce` = 1 or<br>`OpenForce` = 1 and `StopForce` = 1 or<br>`StopForce` = 1 and `CloseForce` = 1 | |
| Forcing and no "local mode" | `OpenForce` = 1 and `CloseForce` = 1 or<br>`OpenForce` = 1 and `StopForce` = 1 or<br>`StopForce` = 1 and `CloseForce` = 1 | |
| "Automatic mode" and no forcing | Pushbutton operation (`Feature.Bit4` = 0):<br>`OpenAut` = 1 and `CloseAut` = 1 or<br>`OpenAut` = 1 and `StopAut` = 1 or<br>`StopAut` = 1 and `CloseAut` = 1<br>Switching mode (`Feature.Bit4` = 1):<br>`OpenAut` = 1 and `CloseAut` = 1 | |
| "Manual mode" and no forcing | `OpenMan` = 1 and `CloseMan` = 1 or<br>`OpenMan` = 1 and `StopMan` = 1 or<br>`StopMan` = 1 and `CloseMan` = 1 | |

## Control system fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 128) section for more on this.

## See also

VlvMotL block diagram (Page 1791)

VlvMotL I/Os (Page 1778)

VlvMotL functions (Page 1762)

VlvMotL modes (Page 1759)

Description of VlvMotL (Page 1754)

## 8.12.5 VlvMotL messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Instance-specific messages

### Process control fault

The following control system error messages can be output:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId1` | SIG 1 | AS process control message - fault | $$BlockComment$$<br>Motor feedback error |
| | SIG 2 | AS process control message - fault | $$BlockComment$$<br>Motor protection triggered |
| | SIG 3 | AS process control message - fault | $$BlockComment$$<br>Valve feedback error |
| | SIG 4 | AS process control message - fault | $$BlockComment$$<br>External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter `CSF`. If it changes to `CSF` = 1, a process control fault is triggered (`MsgEvId1`, SIG 4).

### Instance-specific messages

You can use up to three instance-specific messages with this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId1` | SIG 5 | AS process control message - fault | $$BlockComment$$<br>External message 1 |
| | SIG 6 | AS process control message - fault | $$BlockComment$$<br>External message 2 |
| | SIG 7 | AS process control message - fault | $$BlockComment$$<br>External message 3 |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance `MsgEvId1`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | `ExtVa104` |
| 5 | `ExtVa105` |
| 6 | `ExtVa106` |
| 7 | `ExtVa107` |
| 8 | `ExtVa108` |
| 9 | Reserved |
| 10 | Reserved |

The associated values 4 ... 8 are allocated to the parameters `ExtVa104` ... `ExtVa108` , and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

VlvMotL block diagram (Page 1791)

VlvMotL modes (Page 1759)

## 8.12.6 VlvMotL I/Os

I/Os of VlvMotL

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `AutModLi*` | 1= "Automatic mode" via interconnection or SFC (controlled by `ModLiOp` = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `AutModOp*` | 1 = "Automatic mode" via operator (controlled by `ModLiOp` = 1) | BOOL | 0 |
| `AV` | Input additional analog value, to be connected to `AV_Tech` of the AV block | ANY | |
| `AV_AH_Lim` | Limit high alarm | REAL | 95.0 |
| `AV_AL_Lim` | Limit low alarm | REAL | 5.0 |
| `AV_Hyst` | Hysteresis for alarm, warning and tolerance limits | REAL | 1.0 |
| `AV_TH_Lim` | Limit high tolerance | REAL | 85.0 |
| `AV_TL_Lim` | Limit low tolerance | REAL | 15.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| AV_WH_Lim | Limit high warning | REAL | 90.0 |
| AV_WL_Lim | Limit low warning | REAL | 10.0 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkTypeData* | This input can not be edit or linked and is for internal usage only! | | |
| BypProt | 1 = Bypassing interlock is active in "local mode" and in simulation | BOOL | 0 |
| CloseAut* | 1 = Select Close valve in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CloseChnST | Signal status of output channel for Close<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| CloseForce | 1 = Force valve closure | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CloseLocal | 1 = Select Close valve in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CloseMan* | 1 = Select Close valve in "manual mode" | BOOL | 0 |
| CSF | 1 = External error (control system fault) Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| EN | 1 = Called block will be processed | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| EventTsIn | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16TS block. When this interconnection is configured, the messages of the EventTs, Event16TS block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| EventTs2In | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTs2In input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg3 | Binary input for freely selectable message 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa104 | Associated value 4 for messages (MsgEvID1) | ANY | |
| ExtVa105 | Associated value 5 for messages (MsgEvID1) | ANY | |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| ExtVa108 | Associated value 8 for messages (MsgEvID1) | ANY | |
| FaultExt | 1 = External error<br>Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| FbkClose | 1 = Valve closed feedback signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| FbkClosing | 1 = Valve closing feedback signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| FbkOpen | 1 = Valve open feedback signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| FbkOpening | 1 = Valve opening feedback signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| Feature | I/O for additional functions (Page 1762)<br>1 = Separate monitoring time for stopping the motor | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 13: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0<br>• 0<br>• 0 |
| Feature2 | I/O for additional functions (Page 1762) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| IdleTime* | Wait time for change of direction or restart in [s] | REAL | 5.0 |
| Intlock | 0 = Interlock / interlock Open is activated.<br>Once the interlock condition has cleared, the block can be operated without reset.<br>1 = Interlock / interlock Open is deactivated. | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| Intl_En | 1 = Parameter Intlock is enabled | BOOL | 1 |
| IntlCls | 0 = Interlock Close is activated.<br>Once the interlock condition has cleared, the block can be operated without reset.<br>1 = Interlock Close is deactivated. | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| IntlClsEn | 1 = Parameter IntlCls is enabled | BOOL | 1 |
| LocalLi | 1 = Activate "local mode" via plant signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalOp* | 1 = "Local mode" via operator | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| LocalSetting | Properties for the Local mode (Page 86) | INT | 0 |
| ManModLi* | 1 = Manual mode via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManModOp* | 1= "Manual mode" via: OS operator (controlled via ModLiOp = 0) | BOOL | 1 |
| ModLiOp | Switchover of operating mode between:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MonSafePos | 1 = Go to neutral position in the event of monitoring errors | BOOL | 1 |
| MonTiDynamic* | Monitoring time for feedback errors or feedback start error after successful operation in [s] | REAL | 3.0 |
| MonTiDyStop* | Monitoring time for feedback stop errors after successful operation in [s] | REAL | 3.0 |
| M_Monitor | 1 = Motor feedback monitoring | BOOL | 1 |
| M_MonTiDynamic* | Motor monitoring time after operation in [s] | REAL | 3.0 |
| M_MonTiStatic* | Monitoring time for feedback errors without operation in [s] | REAL | 3.0 |
| MS_RelOp* | 1= Release for maintenance via OS operator | BOOL | 0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_RelOp | 1 = Release for maintenance via OS operator | BOOL | 0 |
| NoFbkClose | 1 = No feedback present for "valve closed" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NoFbkOpen | 1 = No feedback present for "valve open" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = In use by a batch | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OpenAut* | 1 = Select Open valve in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpenChnST | Signal status of output channel for Open<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| OpenForce | 1 = Force valve opening | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpenLocal | 1 = Select Open valve in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpenMan* | 1 = Select Open valve in "manual mode" | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, to be connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 1762)<br>1 = Operator can set or change the monitoring time for "Control: Start"<br>1 = Operator can set or change the monitoring time for "Control: Stop" | STRUCT<br>• Bit 0: BOOL<br>• Bit11: BOOL<br>• Bit 20: BOOL<br>• Bit 30: BOOL | -<br>• 1<br>• 1<br>• 1<br>• 1 |
| OS1Perm | I/O for operator permissions (Page 1762) | STRUCT<br>• Bit 0: BOOL<br>• Bit 15: BOOL | -<br>• 1<br>• 1 |
| Permit | 1 = Activation / activation Open is activated<br>0 = Activation / activation Open is deactivated | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| PermitSafePos | 1=Valve SafePos / Stop allowed | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| PermSf_En | 1=Permit SafePos / Stop enabled | BOOL | 1 |
| Perm_En | 1 = Parameter Permit is enabled | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PermCls | 1 = Activation Close is activated<br>0 = Activation Close is deactivated | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| PermClsEn | 1 = Parameter PermCls is enabled | | |
| Protect | 0 = Protective interlock / protective interlock Open is activated.<br>Once the interlock condition has cleared, the block must be reset.<br>1 = Protective interlock / protective interlock Open is deactivated. | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| Prot_En | 1 = Parameter Protect is enabled | BOOL | 1 |
| ProtCls | 0 = Protective interlock Close is activated.<br>Once the interlock condition has cleared, the block must be reset.<br>1 = Protective interlock Close is deactivated. | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| ProtClsEn | 1 = Parameter ProtCls is enabled | BOOL | 1 |
| PulseWidth* | Pulse width of control signal [s] | REAL | 3.0 |
| RapidStp* | Rapid stop for the motor<br>0 = Motor On<br>1 = Motor Off | BOOL | 0 |
| RstLi* | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | 1 = Reset via operator | BOOL | 0 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SafePos | Neutral position for valve:<br>0 = Closed<br>1 = Open<br>2 = Stop (valve can be in position Closed, Opened or Intermediate) | INT | 2 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SimAV* | Additional value used for SimOn = 1 | REAL | 0.0 |
| SimAV_Li | Additional analog value that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOnLi | 1= Simulation per interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | |
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | |
| StepNo | Batch step number | DWORD | 16#00000000 |
| StopAut* | 1 = Stopping the motor in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopForce | 1 = Force motor stop | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopLocal | 1 = Stopping the motor in "local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopMan* | 1 = Stopping the motor in "manual mode" | BOOL | 0 |
| TorqOpen | 0 = Torque shutdown active when opening<br>1 = "Good" state | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| TorqClose | 0 = Torque shutdown active when closing<br>1 = "Good" state | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| Trip | 1 = Motor is in "good" state | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| UserAna1 | Analog auxiliary value 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UA1unit | Unit of measure for analog auxiliary value 1 | INT | 0 |
| UserAna2 | Analog auxiliary value 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| UA2unit | Unit of measure for analog auxiliary value 2 | INT | 0 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |
| V_Monitor | 1 = Valve feedback monitoring | BOOL | 1 |
| V_MonTiDynamic* | Valve monitoring time after operation in [s] | REAL | 5.0 |
| V_MonTiStatic* | Monitoring time for valve feedback errors without operation in [s] | REAL | 5.0 |
| WarnTiAut* | Prewarning of valve movement from neutral position in "automatic mode" in [s] | REAL | 0.0 |
| StateTrkOn | Initiate tracking of state | BOOL | 0 |
| WarnTiMan* | Prewarning of valve movement from neutral position in "manual mode" in [s] | REAL | 0.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutAct | 1 = "Automatic mode" enabled<br>0 = "Manual mode" is enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutoST | 1 = Worst signal status of automatic commands | BYTE | 16#80 |
| AV_OpScale | Limit for scale in AV bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| AV_Out | Output additional analog value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| AV_Unit | Unit of measure for additional analog value | INT | 0 |
| CascaCut | Cascade connection:<br>1 = Control chain from master controller to secondary valve is interrupted | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Close | Control output<br>1= Close valve | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Closed | 1 = Valve is closed | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Closing | 1 = Valve is closing | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CurrMonV | Remaining valve monitoring time [s] | DINT | 0 |
| CurrMonM | Remaining motor monitoring time [s] | DINT | 0 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see VlvMotL error handling (Page 1774) | INT | -1 |
| FaultExtOut | FaultExt as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkCloseOut | Valve closed feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkClsgOut | Valve closing feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkOpenOut | Valve open feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkOpngOut | Valve opening feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| GrpErr | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IntlockOut | Intlock as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IntlClsOut | IntlCls as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalAct | 1 = "Local mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| LockAct | 1 = Interlock (Intlock, Permit, Protect) or Trip is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManAct | 1 = "Manual mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| MonDynErr | 1 = Feedback error or feedback start error due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| MonDynStopErr | 1 = Feedback stop error due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| M_MonDynErr | 1 = Motor feedback error due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| M_MonStaErr | 1 = Motor feedback error due to unexpected feedback change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message acknowledgment status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | Alarm error 1 (output ERROR of first ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Open | Control output:<br>1 = Open the valve | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Opened | 1 = Valve is open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Opening | 1 = Valve is opening | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1_Perm | WORD | 16#FFFF |
| OS1PermLog | Display of OS1_Perm with settings changed by the block algorithm | WORD | 16#FFFF |
| P_Close | 1 = Pulse signal to close valve | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Open | 1 = Pulse signal to open valve | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Rst | 1= Pulse output for reset<br>The parameter persists for one cycle after a reset. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Stop | 0 = Pulse signal for stopping the valve | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| PermitOut | Permit as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PermitSafePosOut | PermitSafePos as output | STRUCT<br>• Value: BOOL<br>ST: BYTE | -<br>• 1<br>• 16#FF |
| PermClsOut | PermCls as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ProtectOut | Protect as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ProtClsOut | ProtCls as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| R_StpAct | 1 = Rapid stoProtClsOutp of the motor is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RdyToReset | 1 = Ready for reset via RstLi input or commands in "local", "automatic", or "manual mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToStart | 1 = Active start readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 1754) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1754) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1754) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 1754) | DWORD | 16#00000000 |
| Status5 | Status word 5 (Page 1754) | DWORD | 16#00000000 |
| Status6 | Status word 6 (Page 1754) | DWORD | 16#00000000 |
| Stop | 1 = Motor is stopped independent of position | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| TripOut | Trip as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| UA1unitOut | UA1 as output | INT | 0 |
| UA2unitOut | UA2 as output | INT | 0 |
| UserAna1Out | UserAna1 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| UserAna2Out | UserAna2 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| V_MonDynErr | 1 = Valve feedback error due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| V_MonStaErr | 1 = Valve feedback error due to unexpected feedback change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ResyncST | Worst Signal Status of inputs for resynchronization | BYTE | 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| StopCtrl | Control output go to SafePos | BOOL | 0 |
| WarnAct | 1 = Prewarning for valve movement away from neutral position active (parameters WarnTiAut and WarnTiMan) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

**See also**

VlvMotL messaging (Page 1777)

VlvMotL block diagram (Page 1791)

VlvMotL modes (Page 1759)

Error handling (Page 128)

## 8.12.7 VlvMotL block diagram

### VlvMotL block diagram

A block diagram is not provided for this block.

**See also**

VlvMotL I/Os (Page 1778)

VlvMotL messaging (Page 1777)

VlvMotL error handling (Page 1774)

VlvMotL functions (Page 1762)

VlvMotL modes (Page 1759)

Description of VlvMotL (Page 1754)

## 8.12.8 Operator control and monitoring

### 8.12.8.1 VlvMotL views

### Views of the VlvMotL block

The VlvMotL block provides the following views:

- VlvMotL standard view (Page 1792)
- Limit view of motors (Page 328)
- Alarm view (Page 336)

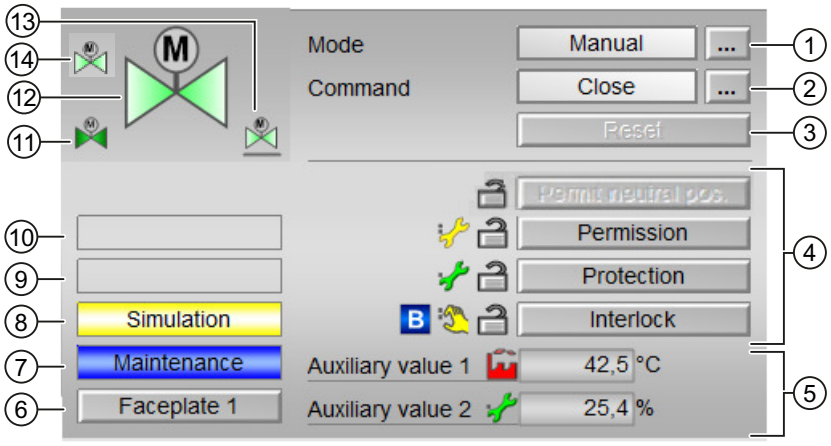
- Trend view (Page 341)
- VlvMotL parameter view (Page 1797)
- VlvMotL preview (Page 1799)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for VlvMotL (Page 1805)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

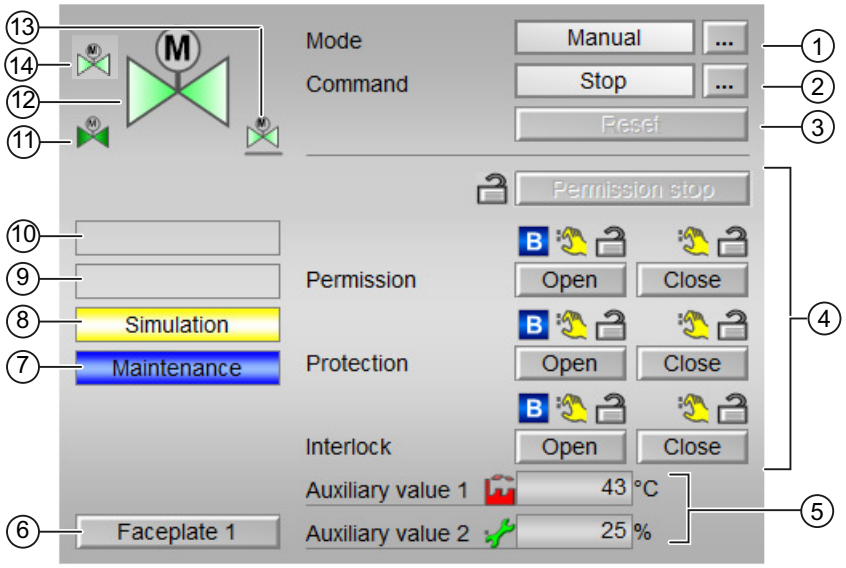
### 8.12.8.2 VlvMotL standard view

#### VlvMotL standard view

With `Feature2.Bit16` = 0:




With `Feature2.Bit16` = 1:




## (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual and automatic mode for motors, valves and dosers (Page 82)
- Manual and automatic mode for motors, valves and dosers (Page 82)
- Local mode (Page 86)
- Out of service (Page 77)

Refer to section Switching operating states and operating modes (Page 289) for information on switching the operating mode.

## (2) Open, close and stop the motor valve

This area shows you the default operating state for the motor valve. The following states can be shown and executed here:

- "Open"
- "Close"
- "Stop"
- "Rapid stop"

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

If the text is configured for these commands, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (3) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in the section Resetting the block in case of interlocks or errors (Page 47).

### (4) Operating range for the interlock functions of the block

This display is only visible when the corresponding block input is connected.

You can use this button to control the interlock functions of the block. You can find additional information on this in the section Interlocking functions (Page 107).

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 113)), for example:

- Signal status (see Forming and outputting the signal status for technologic blocks (Page 117)), for example:

- Bypass information (see Forming the group status for interlock information (Page 113)):

If the text is configured for the interlocks, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (5) Display of auxiliary values

This display is only visible when the corresponding block input is connected.

You can use this area to display two auxiliary values that have been configured in the engineering system. You can find additional information on this in the section Displaying auxiliary values (Page 239).

### (6) Navigation button for switching to the standard view of any faceplate

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234).

### (7) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this is available in section Release for maintenance (Page 70) Display area for block states.

### (8) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"
- "Delay"

You will find more detailed information on this in the sections Simulating signals (Page 64) and Display of delay times (Page 288).

### (9) Display area for block states

This area provides additional information on the operating state of the block:

- "Motor protection"
- "External error"
- "Torque active"
- "End position error"
- "Control error"
- "Internal error"
- "Changeover error"
- "Invalid Signal"

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 104) , Error handling (Page 128) (section "Invalid input signals" and "Mode switchover error") and Motor protection function (Page 106).

### (10) Display area for block states

This area provides additional information on the operating state of the block:

- "Forced open"
- "Forced close"
- "Forced stop"
- "Request 0/1": Reset is expected but reset is not possible from faceplate. For more details, refer:Reset of errors during Request 0/1 (Page 382)

You can find additional information on this in the section Forcing operating modes (Page 45).

### (11) Automatic preview

This display is only visible in "manual mode", in "local mode", or with a reset request in "automatic mode", when the current output signals are not identical to the control in "automatic mode".

The display shows what state the valve would assume if you switched from "manual" or "local" mode to "automatic mode", or performed a reset to "automatic mode".

### (12) Status display of the motor valve

The current status of the motor valve is graphically displayed here.

You can find more information about this in section Block icon for VlvMotL (Page 1805)

### (13) Neutral position of the valve

This representation shows the neutral position for the valve:

- Green: Neutral position is "Open"  or neutral position is “Stop” and valve is opened
- Gray: Neutral position is "Closed" or neutral position is “Stop” and valve is closed
- Light green: Neutral position is "Stop" and valve is in intermediate position

### (14) Mini preview for Resynchronization

Displays the mini preview for Resynchronizing the signal state.

The mini preview displays the following states:

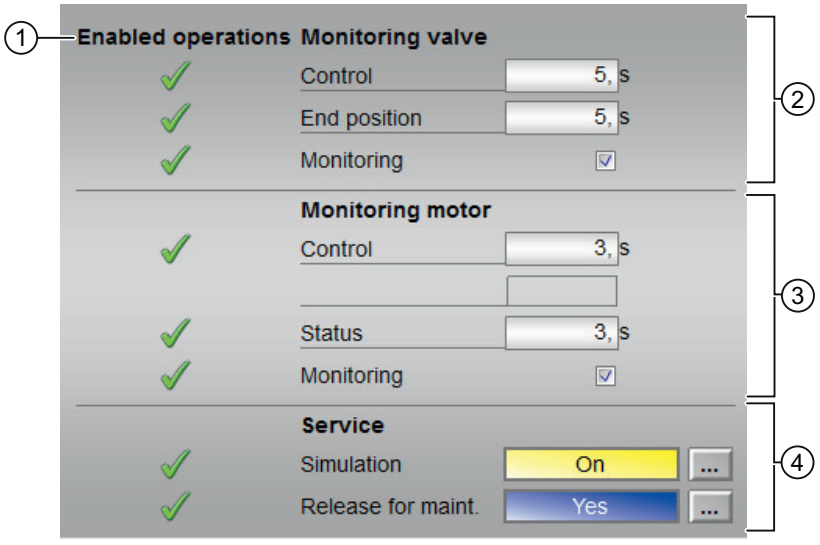
- Open
- Close
- Intermediate

**Note**

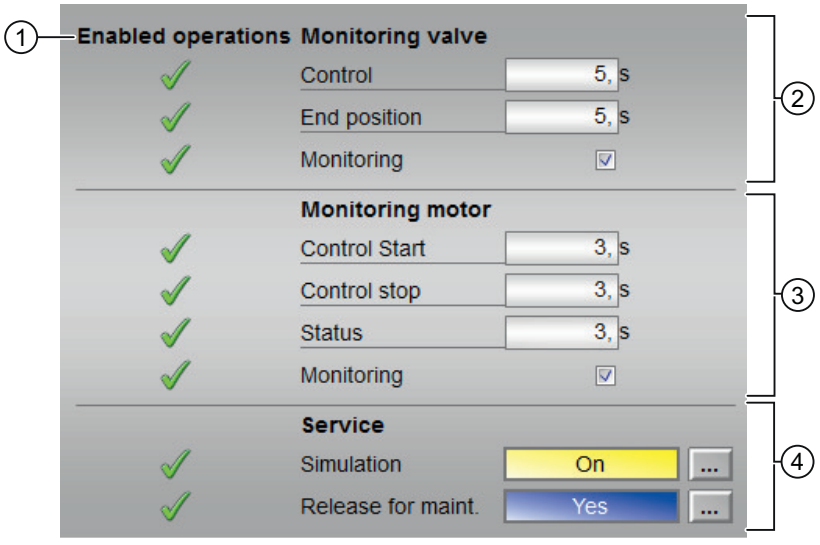
Mini preview for resynch is visible only if resynchronization command is different to control output value.

### 8.12.8.3 VlvMotL parameter view

#### Parameter view of VlvMotL




Parameter view for VlvMotL with `Feature.Bit13` = 0




Parameter view for VlvMotL with `Feature.Bit13` = 1

#### (1) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

### (2) Monitoring valve

In this area, you change parameters and therefore influence the valve. You can find additional information on this in the section Changing values (Page 291).

You can influence the following parameters:

- "Control": Monitoring time for the valve run time (dynamic)
- "End position": Monitoring time for maintaining the valve position (static)
- "Monitoring":
  You can enable monitoring by selecting the check box (☑)
  You can find additional information on this in the section Monitoring the feedbacks (Page 104).

### (3) Monitoring motor

In this area, you change parameters and therefore influence the motor. You can find additional information on this in the section Changing values (Page 291).

You can influence the following parameters:

`Feature.Bit13` **= 0**

- "Control": Monitoring time during startup and shutdown of the motor (dynamic)
- "Status": Monitoring time during permanent operation of the motor (static)

`Feature.Bit13` **= 1**

- "Control start": Monitoring time during startup of the motor (dynamic)
- "Control stop": Monitoring time during shutdown of the motor (dynamic)
- "Status": Monitoring time during permanent operation of the motor (static)
- "Monitoring":
  You can enable monitoring by selecting the check box (☑)
  You can find additional information on this in the section Monitoring the feedbacks (Page 104).

### (4) Service

You can select the following functions in this area:

- "Simulation"
- "Release for maintenance" (with display for a maintenance request)

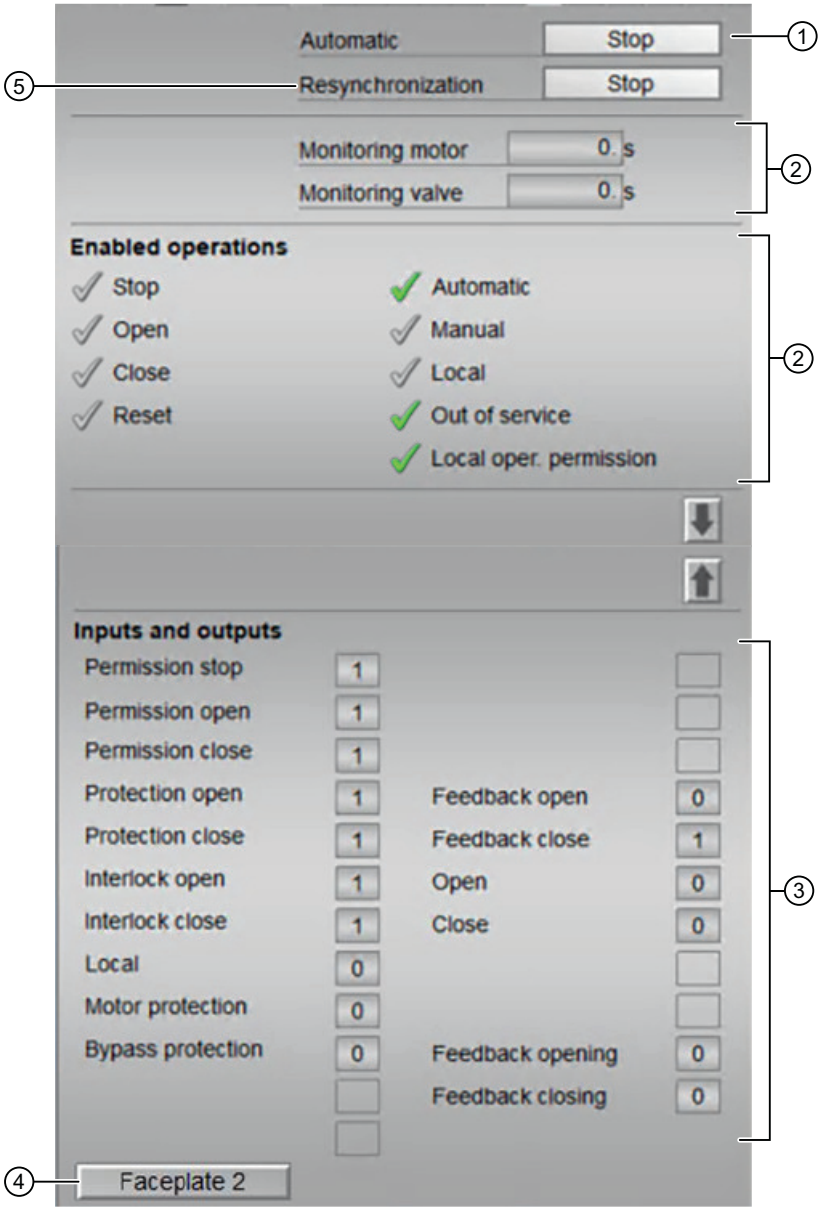
You can find additional information on this in the section Switching operating states and operating modes (Page 289).

You can find information on this area in the section:

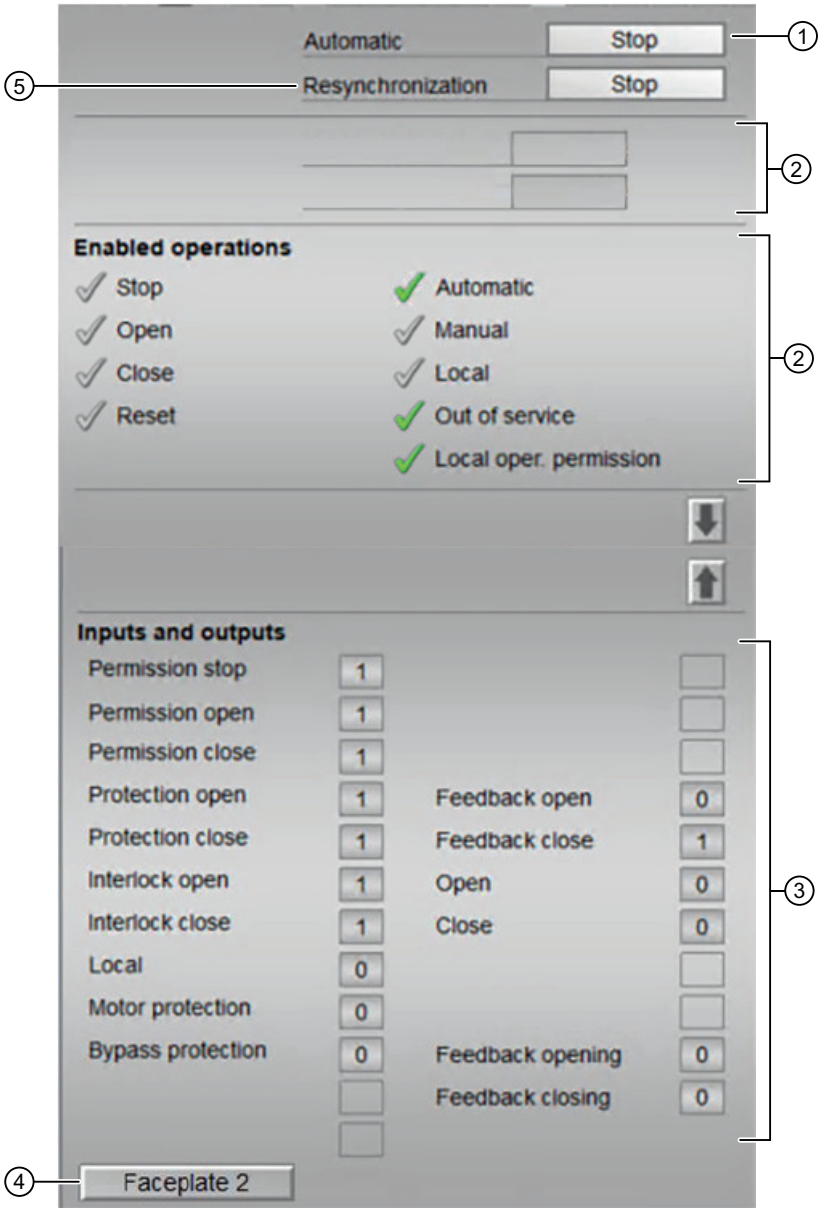
- Simulating signals (Page 64)
- Release for maintenance (Page 70)

### 8.12.8.4 VlvMotL preview

**Preview of VlvMotL**



Display of the current monitoring time of the motor/valve is visible.




Display of the current monitoring time of the motor/valve is not visible.

### (1) Automatic preview

This area shows you the block status after it has switched from the "Manual" mode or "Local" mode to the "Automatic" mode.

If the block is in "Automatic" mode, the current block state is displayed.

**Note**

- If the valve is closed and interlock locking is active (`SafePos`=0 Close), the automatic preview does not show the current status "Closed", instead it shows the active automatic command "Open".
- If the valve is in the end position and the automatic stop command is active, the automatic preview does not show the current status "Closed", but the active automatic command "Stop".

This area also displays the worst signal status of the following automatic commands:

- `OpenAut`
- `CloseAut`
- `StopAut`

### (2) Enable operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "Stop": You can stop the motor of the valve.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Open": You can open the motor valve.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Close": You can close the motor valve.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Reset": You can reset the motor valve if interlocks or errors occur.
- "Automatic": You can switch to "automatic mode".
- "Manual": You can switch to "manual mode".
- "Local": You can switch to "local mode".
- "Out of service": You can switch to "Out of service" operating mode.

- "Local operator permission": Use the ← button to switch to the standard view of the OpStations block. You can find additional information on this in the section Operator control permissions (Page 285).
- "Monitoring motor": Display of the remaining monitoring time of the motor.
- "Monitoring valve": Display of the remaining monitoring time of the valve.

### (3) Displaying current control signals

This area shows the most important parameters for this block with the current selection:

- "Permission" (`Feature2.Bit16` = 0):
  This display is only visible when the corresponding block input is connected.
  - 0 = Motor valve activation not enabled on OS
  - 1 = Enable for "opening"/"closing" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Permission Open" (`Feature2.Bit16` = 1):
  This display is only visible when the corresponding block input is connected.
  - 0 = Motor valve activation not enabled on OS
  - 1 = Enable for "open" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Permission Close" (`Feature2.Bit16` = 1):
  This display is only visible when the corresponding block input is connected.
  - 0 = Motor valve activation not enabled on OS
  - 1 = Enable for "close" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Permit neutral pos" (`Feature2.Bit16` = 0):
  This display is only visible when the corresponding block input is connected.
  - 0 = Prevents the control in safe position direction
  - 1 = prevents the stop command

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection" (`Feature2.Bit16` = 0):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).

- "Protection Open" (`Feature2.Bit16` = 1):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection Close" (`Feature2.Bit16` = 1):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Permit neutral pos" (`Feature2.Bit16` = 0):
  This display is only visible when the corresponding block input is connected.
  - 0 = Prevents the control in safe position direction
  - 1 = prevents the stop command

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock" (`Feature2.Bit16` = 0):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock Open" (`Feature2.Bit16` = 1):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock Close" (`Feature2.Bit16` = 1):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).

- "Local": 1 = Control signal for "Local mode" (LocalLi) is active
- "Motor protection": 1 = Motor is in "good" state
- "Interlock deact.":
  - 0 = Bypass disabled
  - 1 = Bypassing interlock in "local mode" and in "simulation"
- "Torque opening": 0 = Torque shutdown when opening
- "Torque closing": 0 = Torque shutdown when closing
- "Local stop": 1 = Stopping the motor valve in "local mode"
- "Local open": 1 = Opening the motor valve in "local mode"
- "Local close": 1 = Closing the motor valve in "local mode"
- "Feedback open": 1 = Motor valve is open
- "Feedback close": 1 = Motor valve is closed
- "Open": 1 = Motor valve is opened
- "Close": 1 = Motor valve is closed
- "Channel Open": Signal from the output channel block for "Open"
- "Channel Close": Signal from the output channel block for "Close"
- "Feedback opening": 1 = Motor valve is opening
- "Feedback closing": 1 = Motor valve is closing

### (4) Navigation button for switching to the standard view of any faceplate

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

For more information, see section Opening additional faceplates (Page 234).

### (5) Resynchronization

Displays the Resynchronization state and Resynchronization Signal status based on feedback value.

- When motor feedbacks are not present (`Feature.Bit12` = 1) - Resynchronization preview is based on valve feedbacks.
- When motor feedbacks are present (`Feature.Bit12` = 0) - Resynchronization preview is based on motor feedbacks.

### 8.12.8.5 Block icon for VlvMotL

#### Block icons for VlvMotL

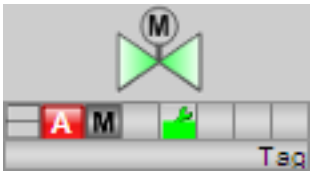
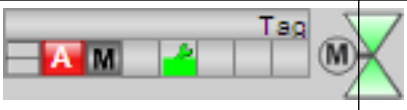
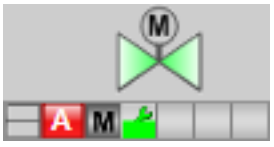
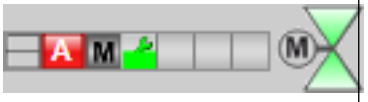
A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as control system faults
- Operating modes
- Signal status, release for maintenance
- Displays for bypassing interlocks
- Interlocks
- Memo display
- Valve status display

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | Non-rotating block icon |
|  | 4 | Non-rotating block icon |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | |
|  | 6 | |
|  | 7 | |
|  | 8 | |
|  | 9 | |

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 10 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

## Valve status display

The following valve states are shown here:

| Icon | Meaning |
|---|---|
|  | Valve open |
|  | Valve closed |
|  | Error at valve |
|  | Valve is opening |
|  | Valve is closing |
|  | Valve stop |
|  | Valve closed |
|  | Valve is closing |

# 8.13 VlvAnL - Control valve

## 8.13.1 Description of VlvAnL

### Object name (type + number) and family

Type + number: FB 1896

Family: Drives

### Area of application for VlvAnL

The block is used for the following applications:

- Control of an analog control valve and positioner with adjustable neutral position
- Control of an optional auxiliary valve for regulating the auxiliary power of the control valve

### How it works

The control valve is brought to a specified position using and analog activation signal. The activation signal can be formed by a ramp function in this case.

The block forms the manipulated variable error from the difference between the activation signal and the acquired position feedback and can monitor it for adherence to high and low limits.

The control valve is monitored for the "Open"/"Closed" position. The block can be connected with a digital limit switch for this purpose. The block can generate the digital position signals itself through the adjustable limits for the "Open"/"Closed" position.

Missing feedback can be derived from the control in the block.

Various inputs are available for control purposes. The next sections provide more detailed information on configuration, operating principles, visualization and operation.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

For the VlvAnL block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Control valve (VlvAnL) (Page 2868)
- Control valve for PA/FF devices (ValveAnalog_Fb) (Page 2869)

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

If `Feature` Bit Set startup characteristics (Page 148) = 0, the following applies to the startup characteristics:

- `Feature.Bit16` = 0 closes the main valve
- With `Feature.Bit16` = 1 the main valve is moved into the neutral position

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section VlvAnL I/Os (Page 1835).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `AutoAct.Value` |
| 6 | `LocalAct.Value` |
| 7 | 0: Open padlock in the block icon<br>1: Closed padlock in the block icon<br><br>In case of `Feature2.Bit16` = 1 then a closed padlock is displayed if Interlock Open OR Interlock Close is active. |
| 8 | Control valve "Open"/"Closed" command (0 = "Closed", 1 ="Open") |
| 9 | `FbkOpenOut.Value` control valve |
| 10 | `FbkCloseOut.Value` control valve |
| 11 | 1 = Feedback error control valve without control change |
| 12 | 1 = Feedback error control valve due to control change |
| 13 | `BypProt` |
| 14 | 1 = Invalid signal status |
| 15 | 1 = Mode changeover error |
| 16 | 1 = `Intlock` is active |
| 17 | 1 = `Permit` is active |
| 18 | 1 = `Protect` is active |
| 19 | Display "Forced open" |
| 20 | Display "Forced stop" |
| 21 | 1 = `Force` |
| 22 | Automatic preview of auxiliary valve (1 = "Open") |
| 23 | 1 = Bumpless switchover to "automatic mode" enabled |
| 24 | 1 = Actuator active (`PosDiClose` < `MV` <`PosDiOpen`) |
| 25 | `UserAna1` interconnected |

| Status bit | Parameter |
|---|---|
| 26 | `UserAna2` interconnected |
| 27 | Auxiliary valve "Open"/"Closed" command (0 =" Closed", 1 = "Open") |
| 28 | `FbkOpenAuxVOut.Value` (auxiliary valve) |
| 29 | `FbkCloseAuxVOut.Value` (auxiliary valve) |
| 30 | 1 = Feedback error auxiliary valve without control change |
| 31 | 1 = Feedback error auxiliary valve due to control change |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | 1 = `MsgLock` message suppression active |
| 1 | 1 = "Open" and "Close" commands deactivated (command line not visible) |
| 2 | 1 = Display for interlocks in block icon |
| 3 | 1 = Display "Opened" for representation of neutral position in `SafePos` = 1 or 2 |
| 4 | 1 = Display "Closed" for representation of neutral position in `SafePos` = 0 or 2 |
| 5 | 1 = Display "Intermediate" for representation of neutral position in `SafePos` = 2 |
| 6 | 1 = Operator can reset the valve |
| 7 | `WarnAct.Value` |
| 8 | 1 = External manipulated variable active (`MV_ExtAct.Value`) |
| 9 | 1 = Forced manipulated variable (`MV_Forced`) is output without limit at output `MV` |
| 10 | 1 = Tracking of manipulated variables `MV`, `MV_TrkOn.Value` = 1 and `MV_ForOn.Value` = 0 |
| 11 | 1 = Manipulated variable greater than limit (low) for manipulated variable `MV` (`MV.Value` > `ManLoLim`) |
| 12 | 1 = Input parameter `Rbk` is not interconnected (`RbkOut.ST` = 16#FF) |
| 13 | 1 = Input parameter `FbkAuxVClose` is connected |
| 14 | 1 = Input parameter `FbkAuxVOpen` is connected |
| 15 | 1 = Input parameter `FbkClose` is connected |
| 16 | 1 = Input parameter `FbkOpen` is connected |
| 17 | For the Error status display in the Closed valve |
| 18 | Reset request in automatic |
| 19 | 1 = No impact of input signals on "local mode" with `LocalSetting` = 0, `LocalSetting` = 2, `LocalSetting` = 4 and OOS mode. |
| 20 | 1 = Control valve open |
| 21 | 1 = Control valve closed |
| 22 | 1 = Control valve opening |
| 23 | 1 = Control valve closing |
| 24 | 1 = Control valve in intermediate position ("Stop") |
| 25 | 1 = Position reached |
| 26 | Control of valve, open auxiliary valve |
| 27 | Control of valve, close auxiliary valve |
| 28 | 1 = Control valve is in intermediate position |
| 29 | For the Error status display in the Opened valve |

| Status bit | Parameter |
|---|---|
| 30 | 1 = Bypass information for display in block icon |
| 31 | `MS_RelOp` |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 | Effective signal 1 of the message block connected via `EventTsIn` |
| 1 | Effective signal 2 of the message block connected via `EventTsIn` |
| 2 | Effective signal 3 of the message block connected via `EventTsIn` |
| 3 | Effective signal 4 of the message block connected via `EventTsIn` |
| 4 | Effective signal 5 of the message block connected via `EventTsIn` |
| 5 | Effective signal 6 of the message block connected via `EventTsIn` |
| 6 | Effective signal 7 of the message block connected via `EventTsIn` |
| 7 | Effective signal 8 of the message block connected via `EventTsIn` |
| 8 | 1 = "Interlock" button is enabled |
| 9 | 1 = "Permission" button is enabled |
| 10 | 1 = "Protection" button is enabled |
| 11 | 1 = Manipulated variable difference high limit violated (`ER_AH_Act.Value`) |
| 12 | 1 = Manipulated variable difference low limit violated (`ER_AL_Act.Value`) |
| 13 | `ER_AH_En` OR `Feature.Bit23` |
| 14 | `ER_AL_En` OR `Feature.Bit23` |
| 15 | 1 = Report manipulated variable difference high limit violation (`ER_AH_MsgEn`) |
| 16 | 1 = Report manipulated variable difference low limit violation (`ER_AL_MsgEn`) |
| 17 | 1 = Readback value high limit violated (`RbkWH_Act.Value`) |
| 18 | 1 = Readback value low limit violated (`RbkWL_Act.Value`) |
| 19 | `RbkWH_En` OR `Feature.Bit23` |
| 20 | `RbkWL_En` OR `Feature.Bit23` |
| 21 | 1 = Report readback value high limit violation (`RbkWH_MsgEn`) |
| 22 | 1 = Report readback value low limit violation (`RbkWL_MsgEn`) |
| 23 | 1 = Automatic preview control valve "Open" |
| 24 | 1 = Automatic preview control valve "Closed" |
| 25 | 1 = Automatic preview control valve "Stop" |
| 26 | 1 = Show automatic preview in the standard view |
| 27 | 1 = Auxiliary valve present |
| 28 | `GrpErr.Value` |
| 29 | `RdyToStart.Value` |
| 30 | `MV_UpRaAct`, `MV_DnRaAct` limits enabled for gradient mode (`MV_RateOn = 1`) |
| 31 | `SimLiOp.Value` |

### Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 | External error generated by FaultExt or external control system fault CSF with set `Feature.Bit18` Activating error state for external process control error CSF (Page 162) |
| 1 | Delay of the `ER_AH_Lim` message |
| 2 | Delay of the `ER_AL_Lim` message |
| 3 | Delay of the `RbkWH_Lim` message |
| 4 | Delay of the `RbkWL_Lim` message |
| 5 | Collection of message delays |
| 6 | 1 = MV ramp active |
| 7 | Resynchronization preview for Open |
| 8 | Resynchronization preview for Close |
| 9 | Resynchronization preview for Stop |
| 10 | Resynchronization: Display preview |
| 11 | Resynchronization: Display small preview symbol in standard view |
| 12 | 1 = Open operation disabled |
| 13 | 1 = Close operation disabled |
| 14 | 1 = Operator MV input in open direction is not allowed |
| 15 | 1 = Operator MV input in close direction is not allowed |
| 16 - 18 | Not used |
| 19 | `Feature2.Bit16`: Separate interlock for each direction, speed or position |
| 20 | Hidden bypass signal in Permit Close |
| 21 | Hidden bypass signal in Interlock Close |
| 22 | Hidden bypass signal in Protect Close |
| 23 | Hidden bypass signal in Permit |
| 24 | Hidden bypass signal in interlock |
| 25 | Hidden bypass signal in Protect |
| 26 | `Feature2.Bit2`: Separate bypass signal |
| 27 | 1 = Input parameter `MV_ChnST` is interconnected |
| 28 | 1 = Input parameter `CtrlChnST` is interconnected |
| 29 | Current monitoring time is visible |
| 30 | Current auxiliary valve monitoring time is visible |
| 31 | Internal error |

### Status word allocation for `Status5` parameter

| Status bit | Parameter |
|---|---|
| 0 - 7 | Effective signal 9..16 of the message block connected via `EventTsIn` |
| 8 - 15 | Not used |
| 16 | "Permission Open" button is enabled |
| 17 | "Permission Close" button is enabled |
| 18 | "Interlock Open" button is enabled |

| Status bit | Parameter |
|---|---|
| 19 | "Interlock Close" button is enabled |
| 20 | "Protect Open" button is enabled |
| 21 | "Protect Close" button is enabled |
| 22 | 1 = Permission Close is active |
| 23 | 1 = Intlock Close is active |
| 24 | 1 = Protect Close is active |
| 25 | 1 = " Permit neutral pos." button is enabled |
| 26 | 1 = Permission SafePos is active |
| 27 | Hidden bypass signal in Permission SafePos |
| 28 | Not used |
| 29 | 1= Suppressed messages |
| 30 | `MV_TrkOn.Value` |
| 31 | Not used |

### See also

VlvAnL modes (Page 1813)

VlvAnL functions (Page 1816)

VlvAnL error handling (Page 1832)

VlvAnL messaging (Page 1833)

VlvAnL block diagram (Page 1850)

## 8.13.2 VlvAnL modes

### VlvAnL operating modes

The block can be operated using the following modes:

- Local mode (Page 86)
- Automatic mode (Page 82)
- Manual mode (Page 82)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

### "Local mode"

The block supports the local modes 2 and 4. Therefore, the control settings for the block are also made based on internal adjustment of the feedback value.

Main valve:

If the readback value is connected /available (`Rbk.ST <> 16#FF`) and the two feedback signals are available the analog main control and the valve position is tracked.

```
MV = Rbk;
```

Valve position open `= (Rbk >= PosDiOpen) AND FbkOpen`

Valve position close `= (Rbk <= PosDiClose) AND FbkClose`

In end position the readback value and the feedback signal are monitored of plausibility.

When readback value is not available only the feedbacks determine the valve position.

When feedback signals are not available only the readback value determines the valve position.

When readback value and feedback signals values are not available the valve position is hold.

Auxiliary valve:

`LocalSetting =2,4`: The control outputs are tracked to the feedbacks.

All two feedbacks are available and plausible means that always only one feedback is active. The control output is tracked:

`Ctrl = NOT (FbkAuxVOpen XOR SafePosAux);` when `FbkAuxVOpen =1, Ctrl = FbkAuxVClose XOR SafePosAux;` when `FbkAuxVClose =1`

When both feedbacks are false then the drive is between of two positions. In this case no monitoring is started. When two feedbacks are active at the same time, then they are not plausible. In this case the static monitoring counter starts.

If one feedback is available it is always plausible.

| NoFbk`AuxV`Open =1 | NoFbk`AuxV`Close =1 |
|---|---|
| `Ctrl = NOT (FbkCloseAuxV XOR SafePosAux)` | `Ctrl = FbkOpenAuxV XOR SafePosAux` |

When all feedbacks are not available the control outputs will be hold.

You can find general information on "Local mode", switching modes and Bumpless switchover in the Local mode (Page 86) section.

## "Automatic mode"

You can find general information on "Automatic mode", switching modes and Bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 82) section.

**With auxiliary valve**

In "automatic mode", the automatic commands affect the auxiliary valve, which you can

- "Open" (`OpenAut` = 1)
- "Close" (`CloseAut` = 1)

**Without auxiliary valve**

In "automatic mode", the automatic commands affect the control valve, which you can

- "Open" (`OpenAut` = 1)
- "Close" (`CloseAut` = 1)

**Note**

If no auxiliary valve is configured, no internal manipulated variable specifications can be made in "automatic mode". Manipulated variable specification is set to external when the mode is switched to "automatic".

## "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the section Manual and automatic mode for motors, valves and dosers (Page 82).

**With auxiliary valve:**

In "manual mode", the operator actions affect the auxiliary valve, which you can:

- "Open" (`OpenMan` = 1)
- "Close" (`CloseMan` = 1)

**Note**

If no auxiliary valve is configured, no external manipulated variable specifications can be made in the manual operating mode. With an internal manipulated value, the switchover from "automatic mode" to "manual mode" is bumpless. With an external manipulated value, the switchover is bumpless only if `MV_TrkExt` = 1 is parameterized.

**Without auxiliary valve:**

In "manual mode", the operator actions affect the control valve, which you can:

- "Open" (`OpenMan` = 1)
- "Close" (`CloseMan` = 1)

**Note**

If no auxiliary valve is configured, no external manipulated variable specifications can be made in the manual operating mode. Manipulated variable specification is set to internal when the mode is switched to manual. The switching is bumpless depending on the parameter `MV_TrkExt`.

## "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature2.Bit1` =0 and `Feature2.Bit1` =1 is as shown below:

| Output Parameters | **`Feature2.Bit1` =0** | **`Feature2.Bit1` =1** |
|---|---|---|
| FbkAuxVOpenOut.ST | 16#80 | 16#60 |
| FbkAuxVCloseOut.ST | 16#80 | 16#60 |
| FbkOpenOut.ST | 16#80 | 16#60 |
| FbkCloseOut.ST | 16#80 | 16#60 |
| FbkOpngOut.ST | 16#80 | 16#60 |
| FbkClsgOut.ST | 16#80 | 16#60 |
| Opened.ST | 16#80 | 16#60 |
| Closed.ST | 16#80 | 16#60 |
| Opening.ST | 16#80 | 16#60 |
| Closing.ST | 16#80 | 16#60 |
| AuxOpened.ST | 16#80 | 16#60 |
| AuxClosed.ST | 16#80 | 16#60 |
| AuxOpning.ST | 16#80 | 16#60 |
| AuxClsing.ST | 16#80 | 16#60 |
| RbkOut.ST | 16#XX | 16#60 |
| RbkWH_Act.ST | 16#XX | 16#60 |
| RbkWL_Act.ST | 16#XX | 16#60 |
| PosReached.ST | 16#80 | 16#60 |
| ER.ST | 16#80 | 16#60 |
| ER_AH_Act.ST | 16#XX | 16#60 |
| ER_AL_Act.ST | 16#XX | 16#60 |

### See also

Description of VlvAnL (Page 1808)
VlvAnL functions (Page 1816)
VlvAnL error handling (Page 1832)
VlvAnL messaging (Page 1833)
VlvAnL I/Os (Page 1835)
VlvAnL block diagram (Page 1850)

## 8.13.3 VlvAnL functions

### Functions of VlvAnL

The functions for this block are listed below.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Interlocks

This block provides the following interlocks:

**`Feature2.Bit16` = 0:**

- Activation enable
- Interlock without reset ("Interlock")
- Interlock with reset ("Protection")
- Permission for safe position ("Permit neutral pos.")

In case of nondirectionally interlock also the auxiliary valve controls to neutral position.

**`Feature2.Bit16` = 1:**

- Activation enable Open
- Activation enable Close
- Interlock open without reset ("Interlock open")
- Interlock close without reset ("Interlock close")
- Interlock open with reset ("Protection interlock open")
- Interlock close with reset ("Protection interlock close")

In case of directionally interlock the auxiliary valve controls do not change.

Refer to the section Interlocks (Page 107) as well as Influence of the signal status on the interlock (Page 112).

### Resetting the block in case of interlocks or errors

This block provides the standard function Resetting the block in case of interlocks or errors (Page 47).

### External error (`FaultExt`), external control system fault (`CSF`)

This block provides the possibility to pass an external error via the `FaultExt` parameter or an external control system fault via the CSF parameter. See, VlvAnL error handling (Page 1832).

### Group error

This block provides the standard function Outputting group errors (Page 130).

The following parameters are taken into consideration when forming the group error:

- `CSF`
- `MonDynErr`
- `MonStaErr`

- MonDynAuxVErr
- MonStaAuxVErr
- FaultExt

### Outputting a signal for start readiness

This block provides the standard function Outputting a signal for start readiness (Page 59).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### Simulating signals

This block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Position feedback (SimRbk, SimRbkLi)

### Using a manipulated variable ramp

The block provides the standard function Using a manipulated variable ramp (Page 133).

This function is ignored with tracking and forced tracking.

### Gradient limiting of the manipulated variable

This block provides the standard function Gradient limiting of the manipulated variable (Page 134).

**Note**

This function is ignored with tracking, forced tracking, forced operating states and travel to the neutral position.

### Tracking and limiting a manipulated variable

You can correct the manipulated variable output to the tracking value MV_Trk or the manipulated variable feedback Rbk to realize Bumpless switchover. To track the manipulated variable output, you have to set the parameter MV_TrkOn = 1.

If the parameter is MV_TrkRbk = 0, the manipulated variable output is corrected to the tracking value MV_Trk. The manipulated variable output MV is limited to the MV_HiLim and MV_LoLim parameters.

If the parameter is MV_TrkRbk = 1, the manipulated variable output is tracked to the position feedback Rbk. There are no limits here.

The "Tracking" text is displayed additionally in the standard view of the faceplate.

### Forming the group status for interlocks

This block provides the standard function Forming the group status for interlock information (Page 113).

### Control loop is open (CascaCut)

The CascaCut output indicates whether the control loop within this block is open. In the open control loop, the master controller must be set to tracking mode to avoid until windup behavior. If you use this block in a cascade structure, then connect the output CascaCut to the input MV_TrkOn of the master controller.

CascaCut is active in the following cases:

- Auxiliary valve is closed
- Block is not in external MV mode
- Tracking function is active
- Force function is active
- Motor protection is active
- Interlock is active
- Monitoring error is active

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The signal status of the output for the manipulated variable feedback RbkOut always corresponds to the signal status of the Rbk input or when the block is in simulation, the signal status of the 16#60 output.

The signal status of the FbkCloseOut and FbkOpenOut outputs is fetched and sent from the worst signal status from the RbkOut output and the corresponding feedback signal inputs FbkClose and FbkOpen.

The signal status of the manipulated variable output MV always corresponds to the signal status of the input parameter MV_Ext or MV_Int, depending on how the setpoint is specified. If the internal manipulated variable MV_Int is used, the signal status is always output as 16#80.

The signal status of manipulated variable difference ER is fetched and sent from the worst signal status of the two outputs RbkOut and MV.

The worst signal status ST_Worst for the block is formed from the following parameters:

- FbkOpenOut.ST
- FbkCloseOut.ST
- FbkAuxVOpenOut.ST
- FbkAuxVCloseOut.ST
- RbkOut.ST
- LocalLi.ST

- `MV_Chn.ST`
- `CtrlChn.ST`
- `OpenAut.ST` (only if `Feature2.Bit10` = 1)
- `CloseAut.ST` (only if `Feature2.Bit10` = 1)
- `MV_Ext.ST` (only if `Feature2.Bit10` = 1)

### Considering bad quality of automatic commands or external values

If the `Feature2` bit Considering bad quality of automatic commands or external values (Page 200) is set to 1 and one of the following parameters has bad signal status (16#00 or 16#28), the block goes to the neutral position in the "Automatic" mode:

- `OpenAut.ST`
- `CloseAut.ST`
- `MV_Ext.ST`

### Forcing operating modes

This block provides the standard function Forcing operating modes (Page 45).

**Note**

If `OpenForce` =1, the high range limit `MV_HiLim` is the output and if `CloseForce` =1, the low range limit `MV_LoLim` is the output that is available at the output `MV`.

If the block is operated with an auxiliary valve, the commands `OpenForce` and `CloseForce` additionally effect the auxiliary valve. With `CloseForce` = 1, the auxiliary valve is closed and the rest position of the control valve is triggered, independent of the output `MV`.

A configured ramp limit has no effect.

### Feedback monitoring

This block provides the standard function Monitoring the feedbacks (Page 104).

Digital feedback signals for the "Open" and "Closed" positions are formed from the position feedback:

- Feedback for "Open" position: `Rbk` ≥ `PosDiOpen`
- Feedback for "Closed" position: `Rbk` ≤ `PosDiClose`

**Specifying control valve positions:**

There are the positions:

- Valve closed ("Closed")
- Valve closes
- Valve open ("Open")

- Valve opens
- Valve has reached defined position

**Valve closed ("Closed")**

When the control valve reaches the "Close" position, the output is `FbkCloseOut.Value` = 1:

- With a binary limit switch for the "Close" position (`NoFbkClose` = 0):
  The valve is considered closed when `FbkCloseOut` is set. `FbkCloseOut` is set when `Rbk` ≤ `PosDiClose` and `FbkClose` = 1.
- Without a binary limit switch for the "Close" position (`NoFbkClose` = 1):
  The valve is considered closed when `FbkCloseOut` is set. `FbkCloseOut` is set when `Rbk` ≤ `PosDiClose`.

**Valve closes**

If the control valve travels in the direction of the "Close" position, the output is `FbkClsgOut = 1`:

- `FbkClsgOut` is set when `MV.Value < RbkOut.Value` and not when `PosReached.Value = 1` .

**Valve open ("Open")**

When the control valve reaches the "Open" position, the output is `FbkOpenOut.Value = 1`:

- With a binary limit switch for the "Open" position (`NoFbkOpen = 0`):
  The valve is considered open when `FbkOpenOut` is set. `FbkOpenOut` is set when `Rbk` ≥ `PosDiOpen` and `FbkOpen = 1`.
- Without a binary limit switch for the "Open" position (`NoFbkOpen = 1`):
  The valve is considered open when `FbkOpenOut` is set. `FbkOpenOut` is set, when `Rbk` ≥ `PosDiOpen`.

**Valve opens**

If the control valve travels in the direction of the "Open" position, the output is `FbkOpngOut` = 1:

- `FbkOpngOut` is set when `MV.Value > RbkOut.Value` and is not `PosReached.Value = 1`.

**Valve has reached defined position**

If the control valve is sent to a specified intermediate position (`MV` > `PosDiClose` and `MV` < `PosDiOpen`), the targeted position is reached when the difference `MV.Value – RbkOut.Value` is within the configured tolerance range ± `PosDeadBand` and therefore `ER.Value = 0.0`.

If the control valve is set to the end position "Open" (`MV.Value >= PosDiOpen`), this position is reached when `ER.Value = 0.0` and also `FbkOpenOut.Value = 1`.

If the control valve is set to the end position "Closed" (`MV.Value <= PosDiClose`), this position is reached when `ER.Value = 0.0` and also `FbkCloseOut.Value = 1`.

When the control valve reaches the specified position, the `PosReached` = 1 output is set.

**Dynamic monitoring**

The monitoring is based on the effective feedback signals `FbkOpenOut` and `FbkCloseOut` and not directly on the feedback inputs `FbkOpen` and `FbkClose`.

A dynamic monitoring error is generated if:

- The "Close" end position is targeted (`MV` ≤ `PosDiClose`) and this position (`FbkCloseOut`) is not reached within the monitoring time `MonTiDynamic`.
- The "Close" end position is targeted (`MV` ≤ `PosDiClose`) and the feedback of the binary limit switch `FbkClose` is already within the control range.
- The "Open" position is targeted (`MV` ≥ `PosDiOpen`) and this position (`FbkOpenOut`) is not reached within the monitoring time `MonTiDynamic`.
- The "Open" end position is targeted (`MV` ≥ `PosDiOpen`) and the feedback for the binary limit switch `FbkOpen` is already within the control range.
- A further intermediate position is controlled and this position is not reached (`PosReached` = 0) within the monitoring time `MonTiDynamic`.

**Static monitoring**

The monitoring is based on the effective feedback signals `FbkOpenOut` and `FbkCloseOut` and not on the feedback inputs `FbkOpen` and `FbkClose`.

A static monitoring error is generated if:

- The "Close" end position (`FbkCloseOut` = 1) is abandoned without a command having been issued beforehand, and the monitoring time `MonTiStatic` has expired.
- The "Close" end position is reached (`FbkCloseOut` = 1), the feedback of the binary limit switch `FbkClose` is gone, and monitoring time `MonTiStatic` has expired.
- The "Open" end position (`FbkOpenOut` = 1) is abandoned without a command having been issued beforehand and the monitoring time `MonTiStatic` has expired.
- The "Open" end position is reached (`FbkOpenOut` = 1), the feedback of the binary limit switch `FbkOpen` is gone, and monitoring time `MonTiStatic` has expired.
- The valve is in intermediate position and with constant `MV`, the analog feedback `Rbk` has been changed so much that the position is no longer reached (`PosReached` = 0) and the monitoring time `MonTiStatic` has expired.

## Suppressing messages using the MsgLock parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

## Monitoring the feedback for the auxiliary valve

The auxiliary valve provides dynamic and static monitoring of the feedback. No special monitoring time can be configured for overseeing the retention of the end position of the auxiliary valve. If the end position is exited without the corresponding command, a "monitoring runtime error" is reported after expiration of the time configured under "Control".

Monitoring is disabled by default and no connections are displayed.

The following parameters can be used to monitor the auxiliary valve

| Parameter | Function |
|---|---|
| FbkAuxVOpen | 1 = Auxiliary valve open feedback signal |
| FbkAuxVClose | 1 = Auxiliary valve closed feedback signal |
| NoFbkAuxVOpen | 1 = No feedback present for Auxiliary valve open (default = 1) |
| NoFbkAuxVClose | 1 = No feedback present for Auxiliary valve closed (default = 1) |
| MonitorAuxV | 1 = Feedback monitoring of the auxiliary valve (default = 0) |
| MonAuxVTime | Monitoring time after auxiliary valve operation in [s] |
| MonDynAuxVErr | Dynamic monitoring error pending |
| MonStaAuxVErr | Static monitoring error pending |

Otherwise, the monitoring works like the monitoring function of the control valve.

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

### Neutral position

This block provides the standard function Neutral position for motors, valves and controllers (Page 54).

The neutral position (de-energized state) for the auxiliary valve is set using the `SafePosAux` parameter.

- `SafePosAux` = 0 means that the auxiliary valve closes with `Ctrl` = 0 and opens with `Ctrl` = 1.
- `SafePosAux` = 1 means that the auxiliary valve opens with `Ctrl` = 0 and closes with `Ctrl` = 1 .

The setting is made with the `SafePos` parameter for the control valve.

- `SafePos` = 0 The control valve is in the "closed" position (`MV.Value` = `MV_LoLim`)
- `SafePos` = 1 The control valve is in the "open" position (`MV.Value` = `MV_HiLim`)
- `SafePos` = 2 The control valve's rest position is "Stop" (`MV.Value` = `RbkOut`, if `Rbk.ST` <> 16 # FF, otherwise MV.Value remains unchanged)

The control valve is brought to the neutral position when the `FbkAuxVCloseOut` = 1 control valve is closed.

### Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 230).

### Disabling of limits

The block provides the standard function Limits disabled in faceplate (Page 206).

## Resynchronization

This block provides the standard function Resynchronization (Page 258).

Resynchronization is switched on over feature bits:

| | |
|---|---|
| Feature2 Bit17 =1 | Resynchronization after CPU startup |
| Feature2 Bit18 =1 | Resynchronization after Simulation or after Out of service |
| Feature2 Bit19 =1 | Resynchronization after control system fault |
| Feature2 Bit20 =1 | Resynchronization with 0->1 trigger at StateTrkOn input |

**Resynch Main Valve**

MV =Rbk; MV is tracked and limited to MV_HiLim and MV_LoLim.

Depending on the feedback input signals and the readback input value the valve state is calculated and set as the feedback output signals**:**

MV =MV_HiLim if FbkOpen And (Rbk >= PosDiOpen)

MV =MV_LoLim if FbkClose And (Rbk <= PosDiClose)

If both feedback signals are not available (NoFbkOpen =1, NoFbkClose =1) only Rbk is considered. If only one feedback and no readback value is available an intermediate position cannot be tracked. Either a feedback signal or a readback value has to be available else resynchronization holds the controls.

**Resynch Auxiliary Valve**

**NoFbkAuxVOpen =0, NoFbkAuxVClose =0:**

**SafePosAux =0:** Ctrl := FbkAuxVOpen AND NOT FbkAuxVClose;

**SafePosAux =1:** Ctrl := FbkAuxVClose AND NOT FbkAuxVOpen;

If there are no plausible feedback positions (FbkAuxVOpen =0, FbkAuxVClose =0 or FbkAuxVOpen =1, FbkAuxVClose =1) the control position is not changed.

**NoFbkAuxVOpen =1, NoFbkAuxVClose =0:**

**SafePosAux =0:** Ctrl := NOT FbkAuxVClose;

**SafePosAux =1:** Ctrl := FbkAuxVClose;

**NoFbkAuxVClose =1, NoFbkAuxVOpen =0:**

**SafePosAux =0:** Ctrl := FbkAuxVOpen;

**SafePosAux =1:** Ctrl := NOT FbkAuxVOpen;

With other combinations **NoFbkAuxVOpen =1, NoFbkAuxVClose =1 (default)** the control position of resynchronization is not changed.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 3 | Enabling resetting of commands for the control settings (Page 172) |
| 4 | Setting switch or button mode (Page 177) |
| 5 | Control via auxiliary valve (Page 182) |
| 6 | Disabling opening and closing (Page 169) |
| 7 | Ramp rate calculation (Page 191) |
| 9 | Resetting via input signals in the event of interlocking (Protection) or errors (Page 174) |
| 10 | Exiting local mode (Page 189) |
| 11 | Activating the run time of feedback signals (Page 164) |
| 12 | Gradient limitation with time duration (Page 197) |
| 15 | Neutral position manipulated variable takes effect with "out of service" operating mode (Page 177) |
| 16 | Neutral position manipulated variable takes effect at startup (Page 176) |
| 17 | Enabling bumpless switchover to automatic mode for valves, motors, and dosers (Page 184) |
| 18 | Activating error state for external process control error CSF (Page 162) |
| 21 | Enable bumpless switchover to "Automatic" mode for operator only (Page 183) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 23 | Limits disabled in faceplate (Page 206) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 27 | Interlock display with LocalSetting 2 or 4 (Page 189) |
| 28 | Disabling operating points (Page 156) |
| 29 | Signaling limit violation (Page 181) |
| 30 | Set reset depending on the operating mode or the LiOp parameter (Page 174) |
| 31 | Activating reset of protection / error in manual mode (Page 175) |

**With auxiliary valve (`Feature.Bit5` = 1):**

In this case the automatic commands `OpenAut` and `CloseAut` control the auxiliary valve

In pushbutton operation (Bit 4 = 0) the automatic commands in automatic mode are latching, in other words `OpenAut`, and `CloseAut`, can be reset to 0 after changing the control. In manual and local modes, however, the automatic commands are not saved and in the absence of automatic commands the automatic control is tracked.

In switching mode (Bit 4 = 1), control is selected with the static signals `OpenAut`. If input `OpenAut` is not set the auxiliary valve is closed. Control via `CloseAut` is not needed. If the

"Activate command reset for control settings" function (Bit 3 = 1) is activated, the `OpenAut` input is reset to the neutral position after evaluation in the block.

**Without auxiliary valve (`Feature.Bit5` = 0):**

The analog valve is opened or closed in automatic mode with the automatic commands `OpenAut` and `CloseAut`. If `OpenAut` and CloseAut are reset, the output MV is set to `MV_Ext`. For "Activate command reset for control settings" (Bit 3 = 1), `OpenAut` and CloseAut are reset and the control elements are self-locking. This means the output `MV` is only set to `MV_Ext` after a change of `MV_Ext`. Control via switch or button mode (bit 4) does not affect the behavior of `OpenAut` and `CloseAut`.

## Configurable reactions using the `Feature2` parameter

| Bit | Function |
|---|---|
| 1 | Changes signal status of outputs in OOS (Page 208) |
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 3 | Control priority in the event of invalid or not plausible input commands (Page 186) |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) |
| 10 | Considering bad quality of automatic commands or external values (Page 200) |
| 11 | Suppress MsgLock and “Out of service” mode for a connected message block (Page 202) |
| 12 | Disable limit act output when limit value message is disabled (Page 209) |
| 16 | Separate interlock for each direction, position, or speed (Page 180) |
| 17 | Resynchronization after startup (Page 213) |
| 18 | Resynchronization after out of service or after simulation (Page 213) |
| 19 | Resynchronization after CSF (Page 214) |
| 20 | Resynchronization with StateTrkOn input (Page 214) |

## Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 239).

## Alarm delays with two time values per limit pair

This block has the standard alarm delay function Two time values per limit pair (Page 227) for limit monitoring of the feedback and limit monitoring of the manipulated variable difference.

The function here relates solely to the limits of the manipulated variable difference.

## Digital feedback from the readback value

This block forms a digital position feedback for "Closed" and "Open" from the configured operating points of the position feedback values `PosDiClose` and `PosDiOpen`.

## Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "automatic mode" |
| 1 | 1 = Operator can switch to "manual mode" |
| 2 | 1 = Operator can switch to "local mode" |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can open the valve |
| 5 | 1 = Operator can close the valve |
| 6 | 1 = Operator can reset the valve |
| 7 | 1 = Operator can define the monitoring time for startup |
| 8 | 1 = Operator can define the monitoring time for the end position |
| 9 | 1 = Operator can enable the monitoring the control valve feedback function (Bit 7 - 8) |
| 10 | 1 = Operator can activate the Simulation function |
| 11 | 1 = Operator can activate the Release for maintenance function |
| 12 | 1 = Operator can change the simulation value `SimRbk` |
| 13 - 23 | Not used |
| 24 | 1 = Operator can define the monitoring time of the auxiliary valve for startup |
| 25 | Not used |
| 26 | 1 = Operator can enable the monitoring the auxiliary valve feedback function (Bit 24) |
| 27 - 31 | Not used |

The block has the following operator permissions for the `OS1Perm` parameter:

| Bit | Function |
|---|---|
| 0 - 3 | Not used |
| 4 | 1 = Operator can switch the manipulated variable to "external" `MV_ExtOp` |
| 5 | 1 = Operator can switch the manipulated variable to "internal" `MV_IntOp` |
| 6 | Not used |
| 7 | 1 = Operator can change the manual manipulated variable `MV_Int` |
| 8 - 9 | Not used |
| 10 | 1 = Operator can change the operation high limit of the manipulated variable `MV_HiLim` |
| 11 | 1 = Operator can change the operation low limit of the manipulated variable `MV_LoLim` |
| 12 | 1 = Operator can enable the manipulated variable's gradient limitation function `MV_RateOn` |
| 13 | 1 = Operator can change the manipulated variable's high limit for the ramp `MV_UpRaLim` |
| 14 | 1 = Operator can change the manipulated variable's low limit for the ramp `MV_DnRaLim` |
| 15 | 1 = Operator can switch between the time value or the value for the ramp (`MV_RmpModTime`) |
| 16 | 1 = Operator can change the ramp time `MV_RmpTime` |
| 17 | 1 = Operator can change the target manipulated variable `MV_RmpTarget` for the manipulated variable ramp |

| Bit | Function |
|---|---|
| 18 | 1 = Operator can enable the manipulated variable ramp function MV_RmpOn |
| 19 | Not used |
| 20 | 1 = Operator can enable the track manipulated variable function in tracking mode MV_TrkRbk |
| 21 | 1 = Operator can enable the bumpless switchover from external to internal MV_TrkExt |
| 22 | 1 = Operator can activate / deactivate messages via ER_AH_MsgEn |
| 23 | 1 = Operator can activate / deactivate messages via ER_AL_MsgEn |
| 24 | 1 = Operator can activate / deactivate messages via RbkWH_MsgEn |
| 25 | 1 = Operator can activate / deactivate messages via RbkWL_MsgEn |
| 26 | 1 = Operator can change the limit (manipulated variable difference) for the high alarm ER_AH_Lim |
| 27 | 1 = Operator can change the hysteresis (manipulated variable difference) ER_Hyst |
| 28 | 1 = Operator can change the limit (manipulated variable difference) for the low alarm ER_AL_Lim |
| 29 | 1 = Operator can change the limit (position feedback) for the high warning RbkWH_Lim |
| 30 | 1 = Operator can change the hysteresis (position feedback) RbkHyst |
| 31 | 1 = Operator can change the hysteresis (position feedback) for low warning RbkWL_Lim |

**Note**

If you interconnect a parameter that is also listed in OS_Perm as a parameter, you have to reset the corresponding OS_Perm bit.

## Generation of manipulated variables

The manipulated variable MV is formed as follows:

| MV_ForOn | CloseForce | OpenForce | ManAct | AutAct | LocalAct | MV_TrkOn | MV_TrkRbk | MV_ExtAct | MV = | Limit | State |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | – | – | – | – | – | – | MV_Forced | none | Forced tracking without limit |
| – | 1 | 0 | – | – | – | – | – | – | MV_OpScale.Low | MV_OpScale.Low | Forced close |
| – | 0 | 1 | – | – | – | – | – | – | MV_OpScale.High | MV_OpScale.High | Forced open |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | 0 | MV_Int | MV_HiLim MV_LoLim | Manual mode with internal manipulated variable |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | – | 1 | MV_ExtOut | MV_HiLim MV_LoLim | Manual mode with external manipulated variable and limit |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | 0 | MV_Int | MV_HiLim MV_LoLim | Automatic with internal manipulated variable and limit |

| MV_ For On | Close Force | OpenF orce | Man Act | Aut Act | Local Act | MV_ Trk On | MV_ Trk Rbk | MV_ Ext Act | MV = | Limit | State |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | – | 1 | MV_ExtOut | MV_HiLim MV_LoLim | Automatic with external manipulated variable and limit |
| 0 | 0 | 0 | – | – | 0 | 1 | 0 | – | MV_Trk | MV_HiLim MV_LoLim | Tracking with limitation |
| 0 | 0 | 0 | 0 | – | 0 | 1 | 1 | – | Rbk | none | Tracking to position feedback without limit |
| 0 | 0 | 0 | 0 | 0 | 1 | – | – | – | Rbk | None | Local mode with tracking to position feedback without limit |

### "Actuator active" information

- The following applies for PosReached.Value = 0:
  With PosDiClose < MV < PosDiOpen, the control valve is detected as active and Bit 24 is set in Status1.
- The following applies for PosReached.Value = 1:
  Status1.Bit 24 = 0

This status can be used to indicate a customized symbol in the process image, for example, and is saved in the status word (see Status word section in Description of VlvAnL (Page 1808)).

### General function "MV difference"

The manipulated variable difference is sent to the ER output and is calculated with the following formula:

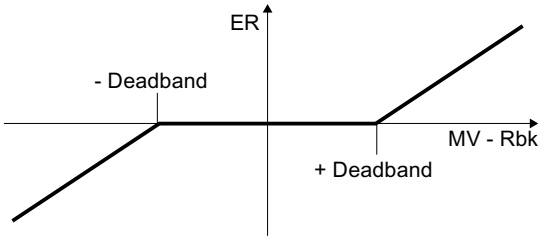
ER = MV - Rbk

If ER is within the dead band, ER = 0 is set and the manipulated variable is considered reached.

### Manipulated variable difference generation and dead band

The manipulated variable difference is formed by the effective manipulated variable MV and the position feedback Rbk, and sent to the ER output. A dead band can be set at the PosDeadBand input:

- PosDeadBand = 0 Dead band is disabled
- PosDeadBand ≠ 0 Dead band is enabled



### Limit monitoring of manipulated variable and error signal

The block provides the standard function Limit monitoring of setpoint, manipulated variable and control deviation (Page 102).

Monitoring is disabled in the following situations:

- The auxiliary valve is closed
- The control valve is in the neutral position

### Providing limits at the output

For further connections to other blocks the following input parameters will also be displayed on the output:

```
ER_AH_LimOut := ER_AH_Lim;
ER_AL_LimOut := ER_AL_Lim;
```

### Group display SumMsgAct for limit monitoring, CSF and ExtMsgx

The block provides the standard function Group display for limit monitoring, CSF and ExtMsgx (Page 92).

### Specifying warning times for control functions at motors and valves

This block provides the standard function Specifying warning times for control functions at motors and valves (Page 56).

The warning time affect the analog manipulated variable `MV`. The output is updated only with the specification of a new manipulated variable after the warning time has expired.

Warning time is ignored in tracking `MV_TrkOn` = 1 and in forced tracking `MV_ForOn`.

### Disabling feedback

This block provides the standard function Disabling feedback for valves (Page 106).

This function is available for both the control valve and auxiliary valve. The feedback for the auxiliary valve is disabled by default, connections are not displayed.

The disable setting is made with the `NoFbkOpen` and `NoFbkClose` parameters for the control valve.

The disable setting is made with the `NoFbkAuxVOpen` and `NoFbkAuxVClose` parameters for the auxiliary valve.

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### Button labels

This block provides the standard function Labeling of buttons and text (Page 235).

Instance-specific text can be configured for the following parameters:

- `OpenMan`
- `CloseMan`

### Outputs for status of drive

The block has the following outputs to indicate the status of the drives. In the CFC plan, this information can be further interconnected:

| | |
|---|---|
| Closed =1 | Valve is in position closed:<br>`MV` <= `PosDiClose`, (`RbkOut` <= `PosDiClose`) AND `FbkClose` =1 |
| Opened =1 | Valve is in position opened:<br>`MV` >= `PosDiOpen`, (`RbkOut` >= `PosDiOpen`) AND `FbkOpen` =1 |
| Closing =1 | Control is in position closed and feedbacks are not in position closed:<br>`MV` <= `PosDiClose`, (`RbkOut` > `PosDiClose`) OR `FbkClose` =0 |
| Opening =1 | Control is in position opened and feedbacks are not in position opened:<br>`MV` >= `PosDiOpen`, (`RbkOut` < `PosDiOpen`) OR `FbkOpen` =0 |
| Stop =1 | Control and feedbacks are in same position. Position is reached. |

**Note**

The outputs for status of drive are only can be active if there is no monitoring error:

`MonDynErr` =0, `MonStaErr` =0

### See also

EventTs functions (Page 2046)

VlvAnL modes (Page 1813)

VlvAnL messaging (Page 1833)

VlvAnL I/Os (Page 1835)

VlvAnL block diagram (Page 1850)

Disable calculation of impulse control in LocalSetting 2 and 4 (Page 200)

## 8.13.4 VlvAnL error handling

### Error handling of VlvAnL

Refer to section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Mode switchover error
- Invalid input signals
- Control system fault (CSF)

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. |
| 41 | The value of the `LocalSetting` connection is outside the valid range. Valid values are 0, 2 and 4 |
| 42 | `LocalSetting` = 0 or `LocalSetting` = 4 and `LocalLi` = 1 |
| 51 | `AutModLi` = 1 and `ManModLi` = 1<br>`OpenAut` = 1 and `CloseAut` = 1<br>`OpenForce` = 1 and `CloseForce` = 1<br>`MV_LiOp` = 1 and `MV_IntLi` = 1 and `MV_ExtLi` = 1; Only when using auxiliary valve (`Feature.Bit5 = 1`) |
| 52 | `LocalSetting` = 2 or 4 and `SimOn` = 1 |

### Mode switchover error

This error can be output by the block, see the section Error handling (Page 128).

### Invalid input signals

This error can be output by the block, see the section Error handling (Page 128).

For the following invalid input signals, the control output can be kept or switched to the neutral position. This depends on the function  control priority for invalid input command (Page 186).

| **Operating mode** | **Invalid input signals** | **Control reaction with `Feature2.Bit3` = 1** |
|---|---|---|
| Forcing | `OpenForce` = 1 and `CloseForce` = 1 | Valve without auxiliary valve (`Feature.Bit5` = 0):<br>Valve is set to its neutral position.<br><br>Valve with auxiliary valve (`Feature.Bit5` = 1):<br>Valve is set to its neutral position. |
| "Automatic mode" and no forcing | Pushbutton operation (`Feature.Bit4` = 0):<br>`OpenAut` = 1 and `CloseAut` = 1 | |
| "Manual mode" and no forcing | `OpenMan` = 1 and `CloseMan` = 1 | |

### Control system fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 128) section for more on this.

### See also

VlvAnL messaging (Page 1833)

Description of VlvAnL (Page 1808)

VlvAnL modes (Page 1813)

VlvAnL functions (Page 1816)

VlvAnL I/Os (Page 1835)

VlvAnL block diagram (Page 1850)

## 8.13.5 VlvAnL messaging

### Messaging

The following messages can be generated for this block:

- Process control fault
- Instance-specific messages
- Process messages

### Process control fault

- The following control system error messages can be output:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 1 | AS process control message - fault | $$BlockComment$$<br>Feedback error |
| | SIG 6 | AS process control message - fault | External error has occurred |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If it changes to CSF = 1, a process control fault is triggered (MsgEvId1, SIG 6).

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 2 | Alarm - high | $$BlockComment$$<br>ER - high alarm limit violated |
| | SIG 3 | Alarm - low | $$BlockComment$$<br>ER - low alarm limit violated |
| | SIG 4 | Warning - high | $$BlockComment$$<br>Rbk - high warning limit violated |
| | SIG 5 | Warning - low | $$BlockComment$$<br>Rbk - low warning limit violated |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Instance-specific messages

You have the option to use one or two instance-specific messages for this block.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG 7 | AS process control message - fault | $$BlockComment$$<br>External message 1 |
| | SIG 8 | AS process control message - fault | $$BlockComment$$<br>External message 2 |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance **MsgEvId1**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |

| Associated value | Block parameters |
|---|---|
| 3 | BatchID |
| 4 | ExtVal04 |
| 5 | ExtVal05 |
| 6 | ExtVal06 |
| 7 | ExtVa107 |
| 8 | ExtVa108 |
| 9 | Reserved |
| 10 | Reserved |

The associated values 4 ... 8 are allocated to the parameters ExtVa104 ... ExtVa108 and can be used. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

### See also

Description of VlvAnL (Page 1808)

VlvAnL functions (Page 1816)

VlvAnL I/Os (Page 1835)

VlvAnL modes (Page 1813)

VlvAnL error handling (Page 1832)

VlvAnL block diagram (Page 1850)

## 8.13.6 VlvAnL I/Os

I/Os of VlvAnL

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameter

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutModLi* | 1= "Automatic mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutModOp* | 1 = "Automatic mode" via operator (controlled by ModLiOp = 0) | BOOL | 0 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BypProt | 1 = Bypassing interlock is active in "local mode" and in simulation | BOOL | 0 |
| CloseAut* | 1 = Select Close valve in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CloseForce | 1 = Force valve closure | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CloseMan* | 1 = Select Close valve in "manual mode" | BOOL | 0 |
| CSF | 1 = External error (control system fault) Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CtrlChnST | Signal status of output channel for Ctrl<br>Should be connected to an output channel block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| EN | 1 = Called block will be processed | BOOL | 1 |
| ER_A_DC* | Delay for incoming alarms during manipulated variable difference monitoring | REAL | 0.0 |
| ER_A_DG* | Delay for outgoing alarms during manipulated variable difference monitoring | REAL | 0.0 |
| ER_AH_En | 1 = Activate alarm (high) for manipulated variable difference monitoring | BOOL | 1 |
| ER_AH_Lim | Alarm limit (high) for manipulated variable difference monitoring | REAL | 100.0 |
| ER_AH_MsgEn | 1 = Activate messages for alarm (high) for manipulated variable difference monitoring | BOOL | 1 |
| ER_AL_Lim | Alarm limit (low) for manipulated variable difference monitoring | REAL | -100.0 |
| ER_AL_En | 1 = Activate alarm (low) for manipulated variable difference monitoring | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ER_AL_MsgEn | 1 = Activate messages for alarm (low) for manipulated variable difference monitoring | BOOL | 1 |
| ER_Hyst | Alarm hysteresis for manipulated variable difference monitoring | REAL | 1.0 |
| EventTsIn | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa104 | Associated value 4 for messages (MsgEvID1) | ANY | |
| ExtVa105 | Associated value 5 for messages (MsgEvID1) | ANY | |
| ExtVa106 | Associated value 6 for messages (MsgEvID1) | ANY | |
| ExtVa107 | Associated value 7 for messages (MsgEvID1) | ANY | |
| ExtVa108 | Associated value 8 for messages (MsgEvID1) | ANY | |
| FaultExt | 1 = External error<br>Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkAuxVClose | 1 = Auxiliary valve closed feedback signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| FbkAuxVOpen | 1 = Auxiliary valve open feedback signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| FbkClose | 1 = Valve closed feedback signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| FbkOpen | 1 = Valve open feedback signal | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#FF |
| Feature | I/O for additional functions (Page 1816) | STRUCT | - |
| | | • Bit 0: BOOL | • 0 |
| | | • ... | • 0 |
| | | • Bit 31: BOOL | • 0 |
| Feature2 | I/O for additional functions (Page 1816) | STRUCT | - |
| | | • Bit 0: BOOL | • 0 |
| | | • ... | • 0 |
| | | • Bit 31: BOOL | • 0 |
| IntlCls | 0 = Interlock Close is activated.<br>Once the interlock condition has cleared, the block can be operated without reset.<br>1 = Interlock Close is deactivated. | BOOL | 1 |
| IntlClsEn | 1 = Parameter IntlCls is enabled | BOOL | 1 |
| Intl_En | 1 = Parameter Intlock is enabled | BOOL | 1 |
| Intlock | 0 = Interlock / Interlock Open is activated.<br>Once the interlock condition has cleared, the block can be operated without reset.<br>1 = Interlock / Interlock Open is deactivated. | STRUCT | - |
| | | • Value:BOOL | • 1 |
| | | • ST:BYTE | • 16#FF |
| LocalLi | 1 = Activate "local mode" via plant signal | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| LocalOp* | 1 = "Local mode" via operator | BOOL | 0 |
| LocalSetting | Properties for the Local mode (Page 86) | INT | 0 |
| ManModLi* | 1 = "Manual mode" via interconnection or SFC (controlled by ModLiOp = 1) | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| ManModOp* | 1= "Manual mode" via: OS operator (controlled via ModLiOp = 0) | BOOL | 1 |
| ModLiOp | Switchover of operating mode between:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| MonAuxVTime* | Monitoring time for feedback monitoring of the auxiliary valve | REAL | 3.0 |
| Monitor | 1 = Monitoring of control valve feedback | BOOL | 1 |
| MonitorAuxV | 1 = Monitoring of auxiliary valve feedback | BOOL | 1 |
| MonSafePos | 1 = Go to neutral position in the event of monitoring errors | BOOL | 1 |
| MonTiDynamic* | Monitoring time after operation in [s] | REAL | 3.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MonTiStatic* | Monitoring time for feedback errors without operation in [s] | REAL | 3.0 |
| MS_RelOp* | 1= Release for maintenance via OS operator | BOOL | 0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_ChnST | Signal status of output channel for MV<br>Should be connected to an output channel block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| MV_DnRaLim* | Gradient limit (low) for manipulated variable [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 100.0 |
| MV_Ext | External manipulated variable | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_ExtLi | Select external manipulated variable (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_ExtOp* | Select external manipulated variable (via operator) | BOOL | 0 |
| MV_Forced | Forced manipulated variable that is not limited and assumes top priority | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_ForOn | 1 = Output forced manipulated variable MV_Forced unlimited at output MV | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_HiLim | Limit (high) for manipulated variable MV | REAL | 100.0 |
| MV_Int* | Internal manipulated variable | REAL | 0.0 |
| MV_IntLi | Select internal manipulated variable (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_IntOp* | Select internal manipulated variable (via operator) | BOOL | 1 |
| MV_LiDnRaLim | Linkable limit (low) for the gradient of the manipulated variable [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 100.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MV_LiOp | Select manipulated variable source (internal/external):<br>1 = Via interconnection<br>0 = Via operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_LiRmpTime | Linkable time [Time unit] for gradient limitation in ramp mode duration<br>Time unit depends on TimeFactor | REAL | 0.0 |
| MV_LiUpRaLim | Linkable gradient limit (high) for the manipulated variable [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 100.0 |
| MV_LoLim | Limit (low) for manipulated variable MV | REAL | 0.0 |
| MV_OpScale | OS display range for manipulated variable MV | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| MV_RateOn* | Change of manipulated variable is limited by MV_UpRaLim and MV_DnRaLim | BOOL | 0 |
| MV_RmpModTime | 1 = Use time values (MV_RmpTime, MV_LiRmpTime)<br>0 = Use gradient limits (MV_DnRaLim, MV_LiDnRaLim, MV_UpRaLim, MV_LiUpRaLim) | BOOL | 0 |
| MV_RmpOn* | 1 = Activate manipulated variable ramp to target value MV_RmpTarget | BOOL | 0 |
| MV_RmpTarget* | Time value for manipulated variable ramp | REAL | 0.0 |
| MV_RmpTime* | Time [Time unit] for manipulated variable ramp from current MV up to MV_RmpTarget or time [Time unit] for gradient limitation in ramp mode duration<br>Time unit depends on TimeFactor | REAL | 0.0 |
| MV_Trk | Tracking value for the manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_TrkExt | 1 = Bumpless switchover from external to internal manipulated variable active | BOOL | 0 |
| MV_TrkOn | 1 = Tracking of manipulated variable MV | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_TrkRbk | 1 = Bumpless switchover of manipulated variable tracking active (track manipulated variable to position feedback) | BOOL | 0 |
| MV_Unit | Unit of measure for manipulated variable | INT | 1342 |
| MV_UpRaLim* | Gradient limit (high) for manipulated variable [Unit/(Time unit)]<br>Time unit depends on TimeFactor | REAL | 100.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| NoFbkAuxVClose | 1 = No "closed" feedback for auxiliary valve present | BOOL | 1 |
| NoFbkAuxVOpen | 1 = No "open" feedback for auxiliary valve present | BOOL | 1 |
| NoFbkClose | 1 = No feedback present for "control valve closed" | BOOL | 0 |
| NoFbkOpen | 1 = No feedback present for "control valve open" | BOOL | 0 |
| Occupied | 1 = In use by a batch | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpenAut* | 1 = Select Open valve in "automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpenForce | 1 = Forced open of control valve | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpenMan* | 1 = Select Open valve in "manual mode" | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for Operator control permissions (Page 285) | STRUCT<br>• Bit 0: BOOL<br>• Bit 12: BOOL<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| OS1Perm | I/O for Operator control permissions (Page 285) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| PermCls | 1 = Activation close enable<br>0 = No activation close enable | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| PermClsEn | 1 = Parameter PermCls is enabled | BOOL | 1 |
| Perm_En | 1 = Parameter Permit enable | BOOL | 1 |
| Permit | 1 = Activation enable / Activation open enable.<br>0 = No activation enable / No activation open enable. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PermitSafePos | 1=Valve SafePos allowed | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL<br>(Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| PermSf_En | 1=Permit SafePos enabled | BOOL | 1 |
| PosDeadBand | Dead band for forming manipulated variable difference | REAL | 0.1 |
| PosDiClose | Limit for control valve position "closed" | REAL | 5.0 |
| PosDiOpen | Limit for control valve position "open" | REAL | 95.0 |
| ProtCls | 0 = Protective interlock close is activated.<br>Once the interlock condition has cleared, the block must be reset.<br>1 = Protective interlock close is deactivated. | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 1<br>• 16#FF |
| ProtClsEn | 1 = Parameter ProtCls enable | BOOL | 1 |
| Prot_En | 1 = Parameter Protect enabled | BOOL | 1 |
| Protect | 0 = Protective interlock / protective interlock open is activated.<br>Once the interlock condition has cleared, the block must be reset.<br>1 = Protective interlock / protective interlock open is deactivated. | STRUCT<br>• Value:BOOL<br>• ST:BYTE | -<br>• 1<br>• 16#FF |
| Rbk | Position feedback for display on OS | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RbkHyst | Alarm hysteresis for position feedback | REAL | 1.0 |
| RbkW_DC* | Delay time for incoming warnings [s] | REAL | 0.0 |
| RbkW_DG* | Delay time for outgoing warnings [s] | REAL | 0.0 |
| RbkWH_En | 1 = Enable warning (high) for position feedback | BOOL | 0 |
| RbkWH_Lim | Limit for position feedback of warning (high) | REAL | 90.0 |
| RbkWH_MsgEn | 1 = Enable messages for warning (high) for position feedback | BOOL | 1 |
| RbkWL_En | 1 = Enable warning (low) for position feedback | BOOL | 0 |
| RbkWL_Lim | Limit for position feedback of warning (low) | REAL | 10.0 |
| RbkWL_MsgEn | 1 = Enable messages for warning (low) for position feedback | BOOL | 1 |
| RstLi* | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | 1 = Reset via operator | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SafePos | Neutral position for control valve:<br>0 = Closed<br>1 = Open<br>2 = Stop (valve can be in position Closed, Opened or Intermediate) | INT | 0 |
| SafePosAux | Neutral position for auxiliary valve:<br>1= Open<br>0 = Closed | BOOL | 0 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | |
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | |
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SimRbk* | Position feedback used for SimOn = 1 | REAL | 0.0 |
| SimRbkLi | Position feedback used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| TimeFactor | Time unit:<br>0 = Seconds<br>1 = Minutes<br>2 = Hours | INT | 0 |
| UA1unit | Unit of measure for analog auxiliary value 1 | INT | 0 |
| UA2unit | Unit of measure for analog auxiliary value 2 | INT | 0 |
| UserAna1 | Analog auxiliary value 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| UserAna2 | Analog auxiliary value 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |
| WarnTiAut | Prewarning of valve movement in automatic mode in [s] | REAL | 0.0 |
| StateTrkOn | Initiate tracking of state | BOOL | 0 |
| WarnTiMan | Prewarning of valve movement in manual mode in [s] | REAL | 0.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutAct | 1 = "Automatic mode" enabled<br>0 = "Manual mode", "Local mode" or "Out of service" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutoST | 1 = Worst signal status of automatic commands | BYTE | 16#80 |
| AuxClosed | 1 = Auxiliary valve is closed | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AuxClsing | 1 = Auxiliary valve is closing | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AuxOpened | 1 = Valve is open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AuxOpning | 1 = Auxiliary valve is opening | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| CascaCut | Cascade connection:<br>1 = Control chain from master controller to secondary valve is interrupted | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| `Closed` | 1 = Valve is closed | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `Closing` | 1 = Valve is closing | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `Ctrl` | Control output for auxiliary valve (depends on `SafePosAuxV`) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `CurrMon` | Remaining monitoring time [s] | DINT | 0 |
| `CurrMonAuxV` | Remaining auxiliary valve monitoring time [s] | DINT | 0 |
| `ENO` | 1 = Block algorithm completed without errors | BOOL | 0 |
| `ER` | Manipulated variable difference | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `ER_AH_Act` | 1 = Alarm limit (high) for manipulated variable difference violated.<br>You can change the reaction for this parameter with `Feature.Bit28` (Disabling operating points (Page 156)) and with `Feature.Bit29` (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `ER_AL_Act` | 1 = Alarm limit (low) for manipulated variable difference violated.<br>You can change the reaction for this parameter with `Feature.Bit28` (Disabling operating points (Page 156)) and with `Feature.Bit29` (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `ER_AH_LimOut` | ER_AH_Lim as output | REAL | 0.0 |
| `ER_AL_LimOut` | ER_AL_Lim as output | REAL | 0.0 |
| `ErrorNum` | Output of pending error number. For error numbers that can be output by this block, see VlvAnL error handling (Page 1832) | INT | -1 |
| `FaultExtOut` | FaultExt as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `FbkAuxVCloseOut` | 1 = Auxiliary valve is closed | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| `FbkAuxVOpenOut` | 1 = Auxiliary valve open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `FbkCloseOut` | 1 = Control valve is open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `FbkClsgOut` | Control valve closing feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `FbkOpenOut` | 1 = Control valve is open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `FbkOpngOut` | Control valve opening feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `GrpErr` | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `IntlockOut` | Intlock as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `IntlClsOut` | IntlCls as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `LocalAct` | 1 = "Local mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `LockAct` | 1 = Interlock (`Intlock`, `Permit` or `Protect`) is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `ManAct` | 1 = "Manual mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| `MonDynAuxVErr` | 1 = Feedback error auxiliary valve due to output change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `MonDynErr` | 1 = Feedback error control valve due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MonStaAuxVErr | 1 = Feedback error from auxiliary valve due to unexpected feedback change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MonStaErr | 1 = Feedback error from control valve due to unexpected feedback change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message acknowledgement status 1 (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | Alarm error 1 (output ERROR of first ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (output STATUS of first ALARM_8P) | WORD | 16#0000 |
| MV | Manipulated variable | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_DnRaAct | Positive limit (low) for manipulated variable is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_ExtAct | 1 = External manipulated variable active<br>0 = Internal manipulated variable active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_ExtOut | Output for external manipulated variable | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_HiAct | 1 = Limit (high) of manipulated variable violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_LoAct | 1 = Limit (low) of manipulated variable violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_RateTarget | Target manipulated variable for the gradient limitation | REAL | 0.0 |
| MV_UnitOut | Unit of measure for manipulated variable | INT | 0 |
| MV_UpRaAct | Positive limit (high) for manipulated variable is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_RemRT | Remaining ramp time of the manipulated variable | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Opened | 1 = Valve is open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Opening | 1 = Valve is opening | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS1PermLog | Display of OS1Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1Perm | DWORD | 16#FFFFFFFF |
| P_Rst | 1= Pulse output for reset<br>The parameter persists for one cycle after a reset. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PermitOut | Permit as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PermitSafePosOut | PermitSafePos as output | STRUCT<br>• Value: BOOL<br>ST: BYTE | -<br>• 1<br>• 16#FF |
| PermClsOut | PermCls as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ProtClsOut | ProtCls as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PosReached | 1 = Control valve has reached specified position | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ProtectOut | Protect as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RbkOut | Position feedback output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RbkWH_Act | 1 = Warning (high) enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RbkWL_Act | 1 = Warning (low) enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToReset | 1 = Ready for reset via RstLi input or commands in "local", "automatic", or "manual mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToStart | 1 = Active start readiness | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 1808) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1808) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1808) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 1808) | DWORD | 16#00000000 |
| Status5 | Status word 5 (Page 1808) | DWORD | 16#00000000 |
| Stop | 1 = Valve is stopped | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| UA1unitOut | UA1 as output | INT | 0 |
| UA2unitOut | UA2 as output | INT | 0 |
| UserAna1Out | UserAna1 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| UserAna2Out | UserAna2 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ResyncST | Worst Signal Status of inputs for resynchronization | BYTE | 16#80 |
| StopCtrl | Control output go to SafePos | BOOL | 0 |
| WarnAct | 1 = Prewarning for control valve movement away from neutral position active (parameters WarnTiAut and WarnTiMan)) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

"local", "automatic", or "manual" mode"

**See also**

VlvAnL modes (Page 1813)

VlvAnL messaging (Page 1833)

VlvAnL block diagram (Page 1850)

Error handling (Page 128)

## 8.13.7 VlvAnL block diagram

### VlvAnL block diagram

A block diagram is not provided for this block.

**See also**

Description of VlvAnL (Page 1808)

VlvAnL modes (Page 1813)

VlvAnL functions (Page 1816)

VlvAnL error handling (Page 1832)

VlvAnL messaging (Page 1833)

VlvAnL I/Os (Page 1835)

## 8.13.8 Operator control and monitoring

### 8.13.8.1 VlvAnL views

#### Views of the VlvAnL block

The VlvAnL block provides the following views:

- VlvAnL standard view with auxiliary valve (Page 1852)
- VlvAnL standard view without auxiliary valve (Page 1857)
- VlvAnL limit view (Page 1863)
- Alarm view (Page 336)
- Trend view (Page 341)
- VlvAnL parameter view (Page 1871)
- VlvAnL preview (Page 1865)
- Memo view (Page 338)
- Batch view (Page 336)
- Ramp view (Page 334)
- Block icon for VlvAnL (Page 1872)

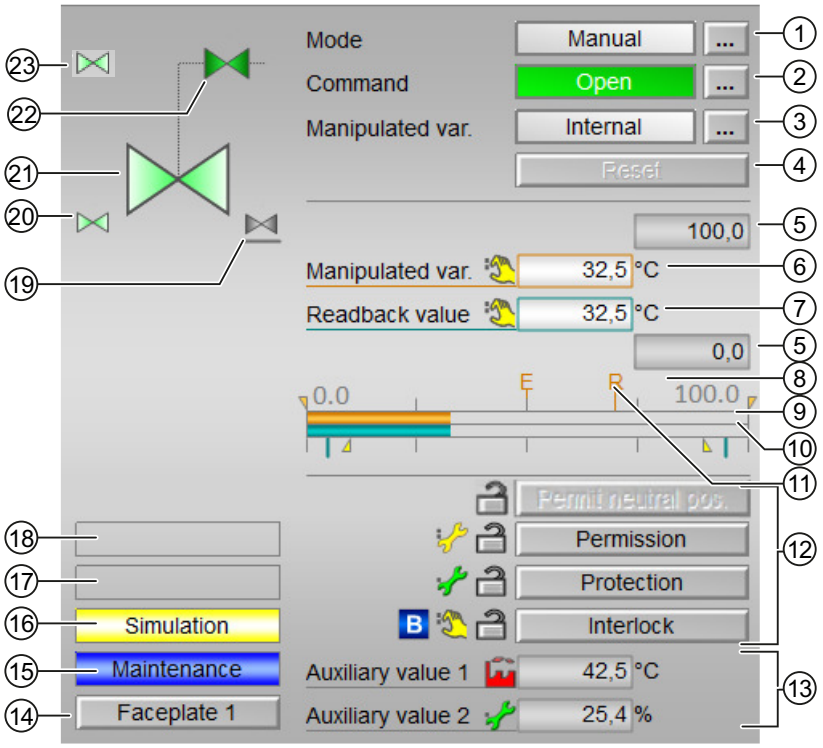
Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

#### See also

Parameter view for motors and valves (Page 320)

### 8.13.8.2 VlvAnL standard view with auxiliary valve

#### Standard view with auxiliary valve of VlvAnL




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual and automatic mode for motors, valves and dosers (Page 82)
- Manual and automatic mode for motors, valves and dosers (Page 82)
- Local mode (Page 86)
- Out of service (Page 77)

Refer to section Switching operating states and operating modes (Page 289) for information on switching the operating mode.

#### (2) Opening, closing and stopping the control valve

This area shows you the default operating state for the valve. The following states can be shown and executed here:

- "Open"
- "Close"
- "Stop" (display only, no operation possible)

"Close"/"open" command relates to the auxiliary valve.

If the text is configured for these commands, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

### (3) Displaying and switching the default manipulated variable

This area shows how to specify the manipulated variable. The manipulated variable can be specified as follows:

- By the application ("External", CFC/SFC)
- By the user directly in the faceplate ("Internal").

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the setpoint specification.

### (4) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in the section Resetting the block in case of interlocks or errors (Page 47).

### (5) High and low scale range for the manipulated variable

These values provide information on the display range for the bar graph of the manipulated variable. The scale range is defined in the engineering system. For more information refer Bar graph (Page 376).

### (6) Displaying and changing the manipulated variable including signal status:

This area shows the current manipulated variable with the corresponding signal status.

The manipulated variable can only be changed by

- Internal manipulated variable specification (`MV_ExtAct` = 0) and
- An opened auxiliary valve (due to the dependency on the standard function Neutral position for motors, valves and controllers (Page 54))

For more information on changing the manipulated variable, refer to section Changing values (Page 291).

### (7) Display of the position feedback including signal status

This area shows the current feedback of the manipulated variable with the corresponding signal status.

### (8) Display for the target manipulated variable of the manipulated variable ramp

This display [R] shows the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 334).

### (9) Bar graph for the manipulated variable

This area shows the current manipulated variable in the form of a bar graph. You can also see the high and low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (10) Bar graph for position feedback

This area shows the current position feedback in the form of a bar graph. You can also see the high and low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

The limits for the "open" and "close" positions are shown with 2 green lines.

### (11) Display of external manipulated variable

This display [E] is only visible when you have selected "Internal" manipulated variable specification. It shows the external manipulated variable that would apply if you were to change the manipulated variable specification to "External".

### (12) Operating range for the interlock functions of the block

This display is only visible when the corresponding block input is connected.

You can use this button to control the interlock functions of the block. You can find additional information on this in the section Interlocking functions (Page 107).

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 113)), for example:

- Signal status (see Forming and outputting the signal status for technologic blocks (Page 117)), for example:

- Bypass information (see Forming the group status for interlock information (Page 113)):

  If there is a bypass, it is displayed instead of the signal status.

If the text is configured for the interlocks, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (13) Display of auxiliary values

This display is only visible when the corresponding block input is connected.

You can use this area to display two auxiliary values that have been configured in the engineering system (ES). You can find additional information on this in the section Displaying auxiliary values (Page 239).

### (14) Navigation button for switching to the standard view of any faceplate

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

See also the section Opening additional faceplates (Page 234) for more on this.

### (15) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this is available in section Release for maintenance (Page 70) Display area for block states.

### (16) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"

You can find additional information on this in the section Simulating signals (Page 64).

### (17) Display area for block states

This area provides additional information on the operating state of the block:

- "External error"
- "End position error"
- "Control error"
- "Internal error"
- "Changeover error"
- "Invalid Signal"

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 104), Error handling (Page 128) (section "Invalid input signals" and "Mode changeover error").

### (18) Display area for block states

This area provides additional information on the operating state of the block:

- "Forced open" (`OpenForce`)
- "Forced close" (`CloseForce`)
- "Forced tracking" (`MV_ForOn`)
- "Tracking" (`MV_TrkOn`)
- "Request 0/1": Reset is expected but reset is not possible from faceplate. For more details, refer: Reset of errors during Request 0/1 (Page 382)
- "MV ramp active"

You can find additional information on this in the section Forcing operating modes (Page 45).

### (19) Representation of neutral position

This representation shows the neutral position for the control valve:

- **Green:** Neutral position is "Open"
- **Gray:** Neutral position is "Closed"
- **Light green:** Neutral position is "Stop"

### (20) Automatic preview

This display is only visible in "manual mode", in "local mode" or with a reset request in "automatic mode", when the current output signals are not identical to the control in "automatic mode".

The display shows what state the valve would assume if you switched from "manual" or "local" mode to "automatic mode" or performed a reset to "automatic mode".

| Mode | Discription |
|---|---|
|  | Open: The control output will go to position open: `MV >= PosDiOpen` |
|  | Close: The control output will go to position close: `MV <= PosDiClose` |
|  | Stop: The control output will go to intermediate position: `PosDiClose < MV < PosDiOpen` |

The `OpenAut`, `CloseAut` commands switch on the energy for the valve.

The symbols are only visible if the new position is different to the actual position.

### (21) Status display of the control valve

You can find more information about this in section Block icon for VlvAnL (Page 1872).

### (22) Picture of auxiliary valve

The small auxiliary valve and the associated line are only shown when the control valve has an additional auxiliary valve and it can be controlled. This can be configured with `Feature.Bit5` on the block.

The current status of the auxiliary valve is graphically displayed here.

### (23) Mini preview for Resynchronization

Displays the mini preview for Resynchronizing the signal state.

The mini preview displays the following states:

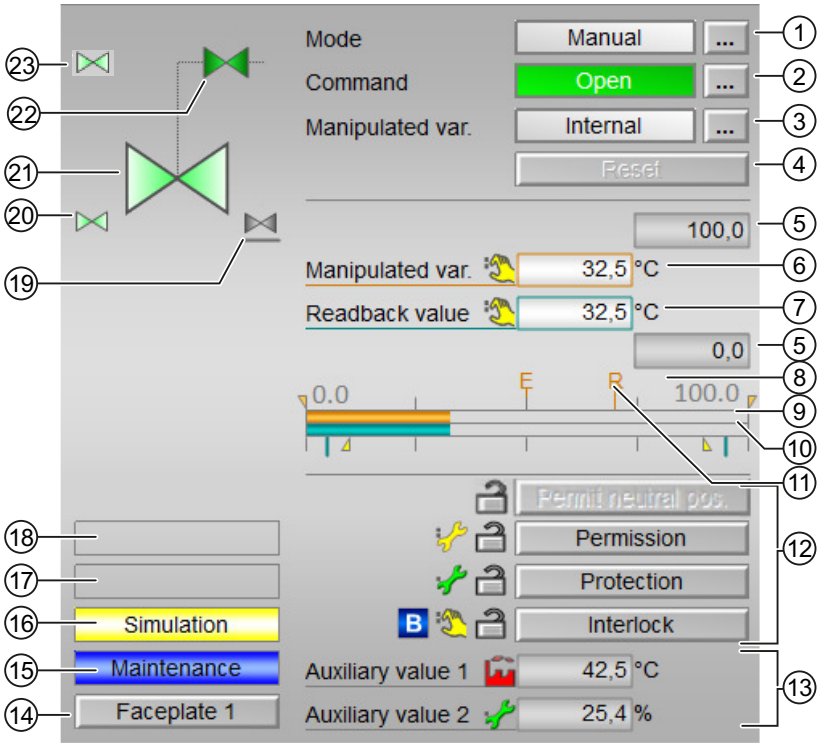
- Open
- Close
- Intermediate

**Note**

Mini preview for resynch is visible only if resynchronization command is different to control output value.

## 8.13.8.3 VlvAnL standard view without auxiliary valve

### Standard view without auxiliary valve of VlvAnL



### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual and automatic mode for motors, valves and dosers (Page 82)
- Manual and automatic mode for motors, valves and dosers (Page 82)
- Local mode (Page 86)
- Out of service (Page 77)

Refer to section Switching operating states and operating modes (Page 289) for information on switching the operating mode.

### (2) Opening, closing and stopping the control valve

This area shows you the default operating state for the valve. The following states can be shown and executed here:

- "Open"
- "Close"
- "Stop" (display only, no operation possible)

"Close"/"Open" command relates to the control valve.

The "Open" command sets the manipulated variable equal to the high limit of the control range (`MV_OpScale.High`).

The "Close" command sets the manipulated variable equal to the low limit of the control range (`MV_OpScale.Low`)

If there is no auxiliary valve (`Feature.Bit5` = 0), the command line can be completely hidden with `Feature.Bit6`.

- 0: (default) line is displayed
- 1: Command line hidden.

If the text is configured for these commands, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

### (3) Displaying and switching the default manipulated variable

This area shows how to specify the manipulated variable. The manipulated variable can be specified as follows:

- By the application ("External", CFC/SFC)
- By the user directly in the faceplate ("Internal").

If there is no auxiliary valve, the specification of the manipulated variable depends on the mode, "manual" or "automatic". You cannot switch between "internal" and "external" here in this case.

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the setpoint specification.

### (4) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in the section Resetting the block in case of interlocks or errors (Page 47).

### (5) High and low scale range for the manipulated variable

These values provide information on the display range for the bar graph of the manipulated variable. The scale range is defined in the engineering system. For more information refer Bar graph (Page 376).

### (6) Displaying and changing the manipulated variable including signal status:

This area shows the current manipulated variable with the corresponding signal status.

The manipulated variable can only be changed by internal manipulated variable specification (`MV_ExtAct` = 0).

For more information on changing the manipulated variable, refer to section Changing values (Page 291).

### (7) Display of the position feedback including signal status

This area shows the current feedback of the manipulated variable with the corresponding signal status.

### (8) Display for the target manipulated variable of the manipulated variable ramp

This display [R] shows the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 334).

### (9) Bar graph for the manipulated variable

This area shows the current manipulated variable in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (10) Bar graph for position feedback

This area shows the current position feedback in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

The limits for the "open" and "close" positions are shown with 2 green lines.

### (11) Display of external manipulated variable

This display [E] is only visible when you have selected "Internal" manipulated variable specification. It shows the external manipulated variable that would apply if you were to change the manipulated variable specification to "External".

### (12) Operating range for the interlock functions of the block

This display is only visible when the corresponding block input is connected.

You can use this button to control the interlock functions of the block. You can find additional information on this in the section Interlocking functions (Page 107).

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 113)), for example:

- Signal status (see Forming and outputting the signal status for technologic blocks (Page 117)), for example:

- Bypass information (see Forming the group status for interlock information (Page 113)):

  If there is a bypass, it is displayed instead of the signal status.

If the text is configured for the interlocks, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (13) Display of auxiliary values

This display is only visible when the corresponding block input is connected.

You can use this area to display two auxiliary values that have been configured in the engineering system (ES). You can find additional information on this in the section Displaying auxiliary values (Page 239).

### (14) Navigation button for switching to the standard view of any faceplate

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

See also the section Opening additional faceplates (Page 234) for more on this.

### (15) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

Additional information on this is available in section Release for maintenance (Page 70) Display area for block states.

### (16) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"
- "Delay"

You will find more detailed information on this in the sections Simulating signals (Page 64) and Display of delay times (Page 288).

### (17) Display area for block states

This area provides additional information on the operating state of the block:

- "External error"
- "End position error"
- "Control error"
- "Internal error"
- "Changeover error"
- "Invalid Signal"

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 104), Error handling (Page 128) (section "Invalid input signals" and "Mode changeover error").

### (18) Display area for block states

This area provides additional information on the operating state of the block:

- "Forced open" (`OpenForce`)
- "Forced close" (`CloseForce`)
- "Forced tracking" (`MV_ForOn`)
- "Tracking" (`MV_TrkOn`)
- "Request 0/1": Reset is expected but reset is not possible from faceplate. For more details, refer: Reset of errors during Request 0/1 (Page 382)
- "MV ramp active"

You can find additional information on this in the section Forcing operating modes (Page 45).

### (19) Representation of neutral position

This representation shows the neutral position for the control valve:

- **Green:** Neutral position is "Open" or neutral position is “Stop” and valve is opened
- **Gray:** Neutral position is "Closed" or neutral position is “Stop” and valve is closed
- **Light green:** Neutral position is "Stop" and valve is in intermediate position

### (20) Automatic preview

This display is only visible in "manual mode", in "local mode" or with a reset request in "automatic mode", when the current output signals are not identical to the control in "automatic mode".

The display shows what state the valve would assume if you switched from "manual" or "local" mode to "automatic mode" or performed a reset to "automatic mode".

| Mode | Discription |
|---|---|
|  | Open: The control output will go to position open: `MV >= PosDiOpen` |
|  | Close: The control output will go to position close: `MV <= PosDiClose` |
|  | Stop: The control output will go to intermediate position: `PosDiClose < MV < PosDiOpen` |

The `OpenAut`, `CloseAut` commands set the MV control to open or close position.

The symbols are only visible if the new position is different to the actual position.

### (21) Picture of control valve

The current status of the control valve is graphically displayed here.

You can find more information about this in section Block icon for VlvAnL (Page 1872).

### (22) Picture of auxiliary valve

The small auxiliary valve and the associated line are only shown when the control valve has an additional auxiliary valve and it can be controlled. This can be configured with `Feature.Bit5` on the block.

The current status of the auxiliary valve is graphically displayed here.

### (23) Mini preview for Resynchronization

Displays the Resynchronization state and Resynchronization Signal status based on feedback and readback values.

The mini preview displays the following states:

- Open
- Close
- Intermediate

**Note**

Mini preview for resynch is visible only if resynchronization command is different to control output value.

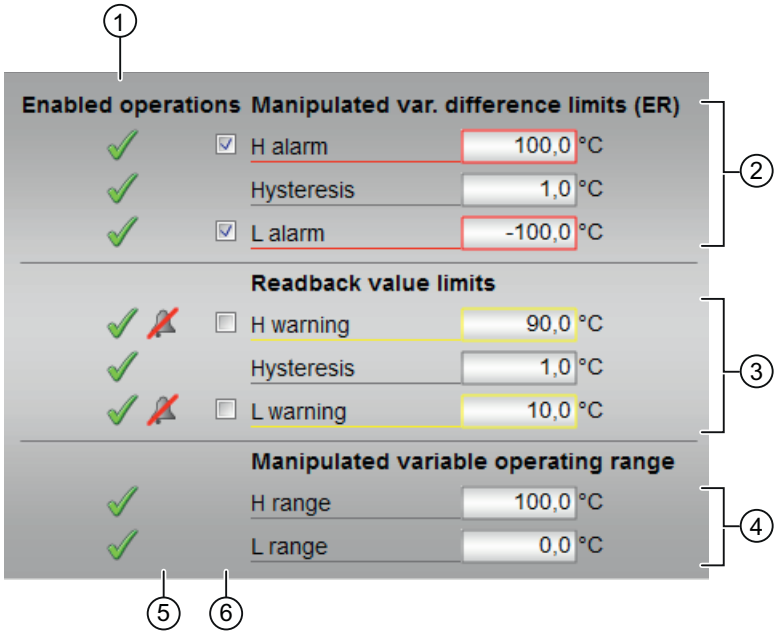
### 8.13.8.4 VlvAnL limit view

#### Limit view of VlvAnL

Several values are set in this view by default:

- Manipulated variable difference limits
- Readback value limits
- Manipulated variable operating range

The toolbars of the faceplate and the block icon indicate when the limits are reached or violated.




#### (1) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

#### (2) Manipulated variable difference limits

In this area, you can enter the limits for the manipulated variable error. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H alarm": Alarm high
- "Hysteresis"
- "L alarm": Alarm low

### (3) Readback value limits (MV)

In this area, you can enter the limits for the readback value (position feedback). Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H warning": Warning high
- "Hysteresis"
- "L warning": Warning low

### (4) Manipulated variable operating range (MV)

In this area, you can enter the limits for the manipulated variable operation range. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H range": Range limit high
- "L range": Range limit low

### (5) Message suppression / delay

Message suppression indicates whether or not the suppression of the associated message in the AS block is activated with the `xx_MsgEn` parameter. The output of messages is not suppressed when the block is installed (all `xx_MsgEn` parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

Alarm delays are also displayed in this position; for more on this see section

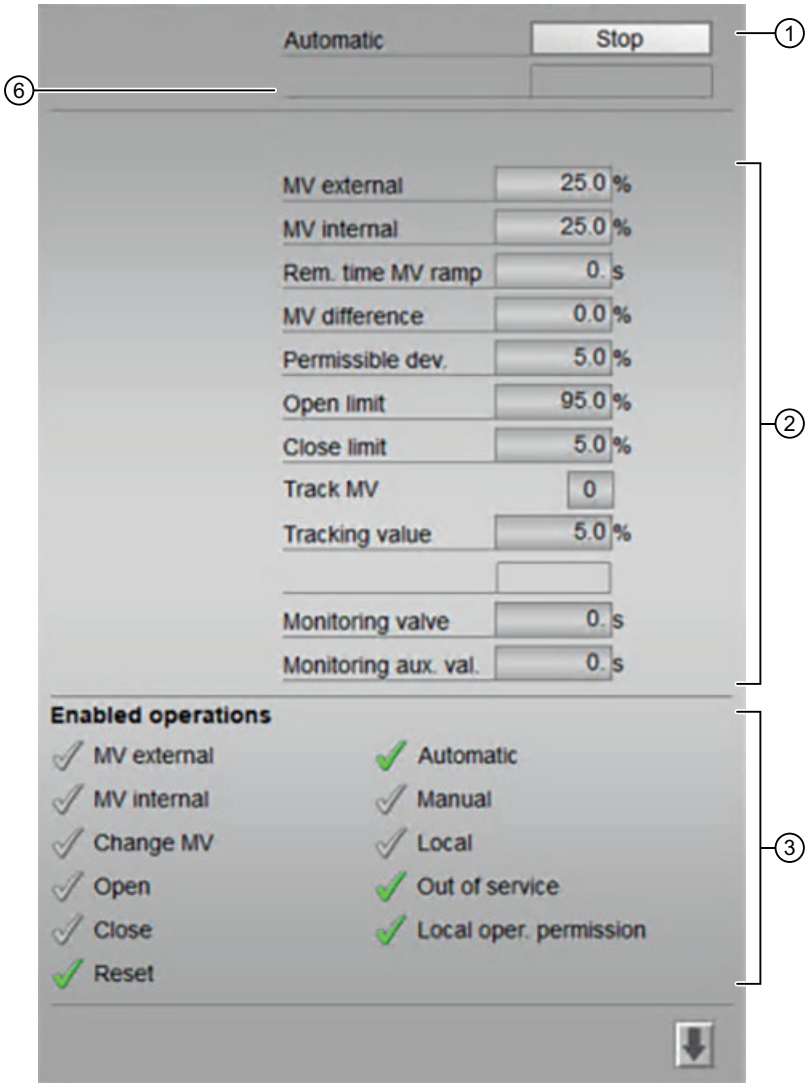
Area of application of the alarm delays (Page 225).

### (6) Suppress messages

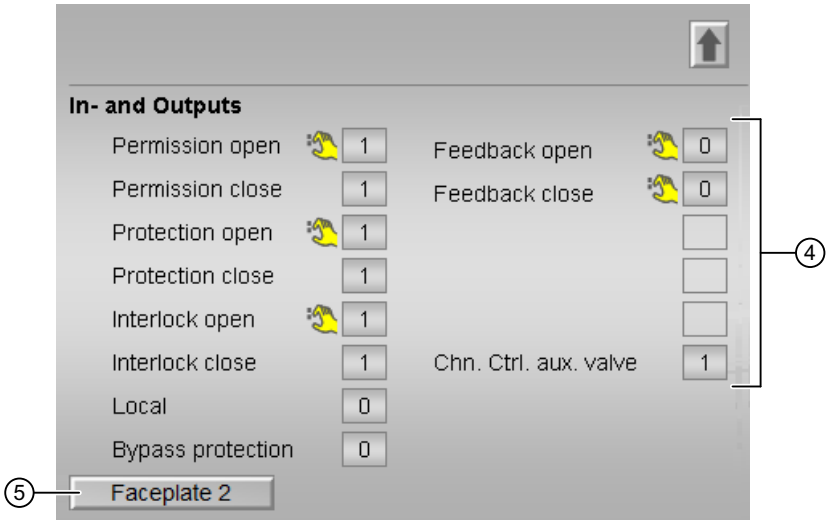
You can enable / disable messages by setting the check mark.

### 8.13.8.5 VlvAnL preview

**Preview of VlvAnL**




The preview has an upper half and a lower half. You can change between the two halves with the arrow keys.




## (1) Automatic preview

This area shows you the block status after it has switched from the "Manual" mode or "Local" mode to the "Automatic" mode.

If the block is in "Automatic" mode, the current block state is displayed.

| Mode | Discription |
|---|---|
| Open | Automatic commands will open the valve or valve is already opened: `MV >= PosDiOpen` |
| Close | Automatic commands will close the valve or valve is already closed: `MV <= PosDiClose` |
| Stop | Automatic commands will set the valve to intermediate position or valve is already in intermediate position: `PosDiOpen <= MV <= PosDiClose` |

**Note**

If the valve is closed and interlock locking is active (SafePos=0 Close), the automatic preview does not show the current status "Closed", instead it shows the active automatic command "Open".

This area also displays the worst signal status of the following automatic commands:

- `OpenAut`
- `CloseAut`
- `MV_Ext`

### (2) Preview area

- "Manipulated variable external": Display current external manipulated variable (`MV_ExtOut`).
- "Manipulated variable internal": Display current internal manipulated variable (`MV_Int`).
- "Rem. time MV ramp": Remaining time to reach the ramp target value.
- "Manipulated variable difference": Current manipulated variable error (`ER`)
- "Permissible dev.": Permissible ± deviation (`PosDeadBand`) of manipulated variable that is output. If the manipulated variable feedback `Rbk` is within this range, the manipulated variable is considered reached.
- "Open limit": Limit (`PosDiOpen`) for forming the "Control valve Open" signal (`FbkOpenOut`). If the position feedback reaches this limit, the control valve is open.
- "Close limit": Limit (`PosDiClose`) for forming the "Control valve Closed" signal (`FbkCloseOut`). If the position feedback reaches this limit, the control valve is closed.
- "Manipulated variable tracking": (`MV_TrkOn` = 1) manipulated variable is corrected to the tracking value. Tracking value for the effective manipulated variable for "Track manipulated variable"
- "Channel MV" Display of the manipulated variable by the output channel block
- "Auxiliary valve monitoring" Display of the remaining monitoring time of the auxiliary valve.
- "Monitoring valve" Display of the remaining monitoring time of the valve.
- "Rbk Resync": Shows the input Rbk value for Resynchronization

### (3) Enable operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "Manipulated variable external": You can enable external manipulated variable specification
- "Manipulated variable internal": You can enable internal manipulated variable specification
- "Change MV": You can change the manipulated variable
- "Open": You can open the valve. The display always relates to the auxiliary valve and control valve. Neither the auxiliary valve nor the control valve can be opened if there is no enable. For the control valve this means: If a new manipulated variable is greater than the current valve position, this new manipulated variable take effect.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).

- "Close": You can close the valve. The display always relates to the auxiliary valve and control valve. Neither the auxiliary valve nor the control valve can be closed if there is no enable. For the control valve this means: If a new manipulated variable is less than the current valve position, this new manipulated variable does not take effect and the control valve retains its position.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Reset": You can reset the valve if interlocks or errors occur.
- "Automatic": You can switch to "automatic mode".
- "Manual": You can switch to "manual mode".
- "Local": You can switch to "local mode".
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operator permission": Use the ← button to switch to the standard view of the OpStations block. For more information, see section Operator control permissions (Page 285).

### (4) Inputs and outputs

This area shows the most important parameters for this block with the current selection.

- "Permission" (`Feature2.Bit16` **= 0**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Motor valve activation not enabled on OS
  - 1 = Enable for "opening"/"closing" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Permission open" (`Feature2.Bit16` **= 1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Motor valve activation not enabled on OS
  - 1 = Enable for "open" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Permission close" (`Feature2.Bit16` **= 1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Motor valve activation not enabled on OS
  - 1 = Enable for "close" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).

- "Permit neutral pos" (`Feature2.Bit16` = 0):
  This display is only visible when the corresponding block input is connected.
  - 0 = prevents stop command to the motor
  - 1 = prevents the control in safe position direction

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection" (`Feature2.Bit16` **= 0**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is in effect; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection open" (`Feature2.Bit16` **= 1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is in effect; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection close" (`Feature2.Bit16` **= 1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is in effect; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock" (`Feature2.Bit16` **= 0**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock open" (`Feature2.Bit16` **= 1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).

- "Interlock close" (`Feature2.Bit16` **= 1**):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Local": 1 = Control signal for "Local mode" (LocalLi) is active
- "Interlock deact.":
  - 0 = Bypass disabled
  - 1 = Bypassing interlock in "local mode" and in "simulation"

Since, depending on the configuration, the block itself forms the digital feedback signals internally via the "Open" and "Closed" positions, the following reaction results from the "Feedback open" and "Feedback closed" signals.

- Control valve:
  - "Feedback open": The display is derived from the `FbkOpenOut` output. However, this output is formed from the configured limit for the "Open" position.
  - "Feedback close": The display is derived from the `FbkCloseOut` output. However, this output is formed from the configured limit for the "Closed" position.
- "Control auxiliary valve": only visible when there is a auxiliary valve
  - Control binary control `Ctrl.Out`
  - "Feedback open auxiliary valve": `FbkAuxVOpenOut`.
  - "Feedback closed auxiliary valve": `FbkAuxCloseOut`.
  - "Channel Control auxiliary valve": Signal for controlling the auxiliary valve by the output channel block

### (5) Navigation button for switching to the standard view of any faceplate

This display is only visible when the corresponding block input is connected.

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

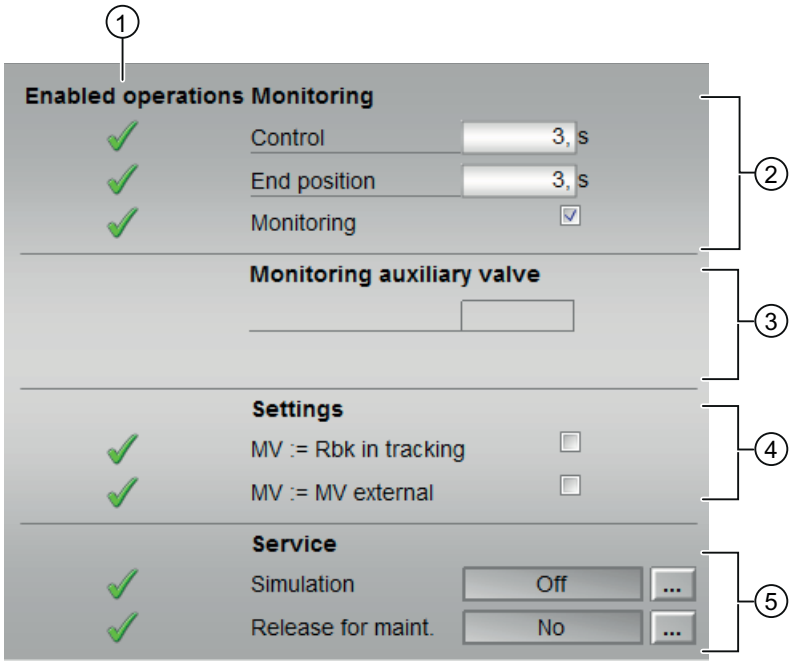
For more information, see section Opening additional faceplates (Page 234).

### (6) Resynchronization

Displays the Resynchronization state and Resynchronization Signal status based on feedback and readback values.

### 8.13.8.6 VlvAnL parameter view

#### Parameter view of VlvAnL




#### (1) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

#### (2) Monitoring

In this area, you change parameters and therefore influence the control valve. Refer to the Changing values (Page 291) section for more on this.

You can influence the following parameters:

- "Control": Monitoring time during opening and closing of the control valve (dynamic)
- "End position": Monitoring time of the end position of the control valve (static)

**Enable monitoring**

You can enable monitoring by selecting the check box (☑)

You can find additional information on this in the section Monitoring the feedbacks (Page 104).

### (3) Monitoring auxiliary valve

As under **(2)** Monitoring, but only visible when there is an auxiliary valve. Only one common monitoring time can be entered for dynamic and static monitoring.

### (4) Settings

- `MV` = `Rbk` in tracking mode: Correction is performed with the position feedback `Rbk` instead of the `MV_Trk` tracking value. The switchover from "Track manipulated variable" (`MV_TrkOn` = 1) to "Do not track manipulated variable" is bumpless (`MV_TrkOn` = 0).
  Set `MV_TrkRbk` parameter.
- `MV` = `MV` extern: Bumpless switchover of manipulated variable from external to internal. The internal manipulated variable is tracked to the external one.
  Set `MV_TrkExt` parameter.

### (5) Service

You can select the following functions in this area:

- "Simulation"
- "Release for maintenance" (with display for a maintenance request)

Refer to the Switching operating states and operating modes (Page 289) section for more on this.

You can find information on this area in the section:

- Simulating signals (Page 64)
- Release for maintenance (Page 70)

## 8.13.8.7 Block icon for VlvAnL

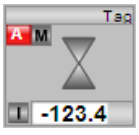
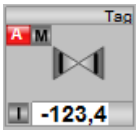
### Block icons for VLvAnL

A variety of block icons are available with the following functions:

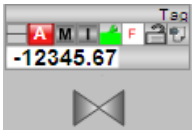
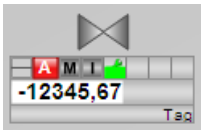
- Limits (high/low)
- Alarm violation. Warning and tolerance limits as well as control system faults
- Operating modes
- Signal status, release for maintenance
- Forcing states
- Displays for bypassing interlocks

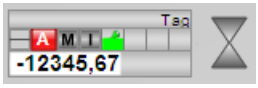
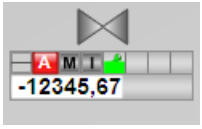
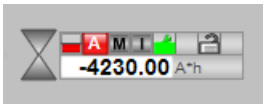
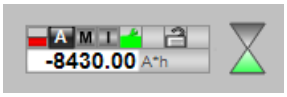
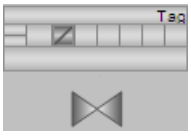
- Interlocks
- Valve status display
- Display of feedback value (white, with decimal places)
- Operation of the manipulated variable
- Process tag type
- External/internal manipulated variable specification
- Memo display

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | Non-rotating block icon |
|  | 4 | Non-rotating block icon |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | 3 | |

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 4 | |
|  | 5 | |
|  | 6 | |
|  | 7 | |
|  | 8 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

## Valve status display

The following valve states are shown here:

| Icon | Meaning |
|---|---|
|  | Valve open |
|  | Valve closed |
|  | Error at valve |
|  | Valve is opening |

| Icon | Meaning |
|---|---|
|  | Valve is closing |
|  | Valve stopped |
|  | Valve closed |
|  | Valve is closing |

# 8.14 VlvPosL - Valve positioner

## 8.14.1 Description of VlvPosL

### Object name (type + number) and family

Type + number: FB 1918

Family: Drives

### Area of application for VlvPosL

The block is used for the following application:

- Control of a motor valve with analog position feedback.

### How it works

Various operating modes are available for controlling the motorized valve. This enables you to change the valve states individually. All changes of modes or states, as well as errors occurring in this context, are monitored, visualized in the faceplate and reported to the operator. Operators with suitable permissions can use the block icon and the faceplate to view the current state of the motorized valve and operate it.

The motor valve with analog position feedback can be controlled via binary signals such as "Open", "Close", or "Stop" or via analog manipulated variable signals. An internal position controller is activated when analog manipulated variable signals are used. This enables even motor valves with delayed reaction in their position feedback to be position controlled.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

Following startup, the messages are suppressed for the number of cycles assigned in the `RunUpCyc` parameter.

After a startup without control (`Open, Close` = 0) no monitoring of the feedback signals `FbkOpen` and `FbkClose` takes places during the `V_MonTiStatic` time. Changes to `FbkOpen` and `FbkClose` are applied. This means that the feedback is monitored again even in the stop state.

### Status word assignment for `Status1` parameter

You can find a description of the individual parameters in the section VlvPosL I/Os (Page 1912).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `AutoAct.Value` |
| 6 | `LocalAct.Value` |
| 7 | 0 = Open padlock in the block icon<br>1 = Closed padlock in block icon<br>With `Feature2.Bit16` = 1, a closed padlock is displayed if "Interlock Open" or "Interlock Close" is active. |
| 8 | `Open.Value` |
| 9 | Motor is stopped |
| 10 | `Close.Value` |
| 11 | Torque shutoff enabled (`TorqOpen` or `TorqClose` = 1)<br>When the seal valve function is enabled, the torque shutoff is only active when the `TorqClose` signal is active before the valve feedback. |
| 12 | `WarnAct.Value` |
| 13 | Feedback error without control change |
| 14 | Feedback error due to control change |
| 15 | Mode switch fail |
| 16 | 1 = `Intlock` is active |
| 17 | 1 = `Permit` is active |
| 18 | 1 = `Protect` is active |
| 19 | Displays forced operating state in the block icon: Force |
| 20 | Not used |
| 21 | 1 = Position control is active |
| 22 | 1 = Position control is not active |
| 23 | "Interlock" button is enabled |
| 24 | 0 = Display neutral position "Closed" Display "Closed" for representation of neutral position in `SafePos` = 0 or 2<br>1 = Display "Opened" for representation of neutral position in `SafePos` = 1 or 2 |
| 25 | 1 = Display "Intermediate" for representation of neutral position in `SafePos` = 2 |
| 26 | Bypass information for display in block icon |
| 27 | Bypass active (`BypProt` = 1) and `Local.Act` = 1 or `SimOn` = 1 |
| 28 | Invalid signal status |
| 29 | 0 = Closed<br>1 = Open |
| 30 | "Permission" button is enabled |
| 31 | "Protection" button is enabled |

### Status word assignment for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `MsgLock` |
| 1 | `AV_AH_Act.Value` |
| 2 | `AV_WH_Act.Value` |
| 3 | `AV_TH_Act.Value` |
| 4 | `AV_TL_Act.Value` |
| 5 | `AV_WL_Act.Value` |
| 6 | `AV_AL_Act.Value` |
| 7 | `AV_AH_En` OR `Feature.Bit23` |
| 8 | `AV_WH_En` OR `Feature.Bit23` |
| 9 | `AV_TH_En` OR `Feature.Bit23` |
| 10 | `AV_TL_En` OR `Feature.Bit23` |
| 11 | `AV_WL_En` OR `Feature.Bit23` |
| 12 | `AV_AL_En` OR `Feature.Bit23` |
| 13 | `AV_AH_MsgEn` |
| 14 | `AV_WH_MsgEn` |
| 15 | `AV_TH_MsgEn` |
| 16 | `AV_TL_MsgEn` |
| 17 | `AV_WL_MsgEn` |
| 18 | `AV_AL_MsgEn` |
| 19 | 1 = No impact of input signals on "local mode" with `LocalSetting` = 0, `LocalSetting` = 2, `LocalSetting` = 4 and OOS mode. |
| 20 | 1 = Valve is closing |
| 21 | 1 = Valve is closed |
| 22 | 1 = Valve is stopped |
| 23 | 1 = Valve is opening |
| 24 | 1 = Valve is open |
| 25 | For the error status display in the closed valve |
| 26 | For the error status display in the opened valve |
| 27 | Automatic preview for "Open" |
| 28 | Automatic preview for "Close" |
| 29 | Automatic preview for "Stop" |
| 30 | Display for interlocks in block icon |
| 31 | `MS_RelOp` |

### Status word assignment for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 | `M_MonStaErr.Value` |
| 1 | `M_MonDynErr.Value` |
| 2 | `V_MonStaErr.Value` |
| 3 | `V_MonDynErr.Value` |

| Status bit | Parameter |
|---|---|
| 4 | M_MonStopErr.Value |
| 5 | Automatic preview for "Position Control" on |
| 6 | Automatic preview for "Position Control" off |
| 7 | Torque shutoff opening enabled |
| 8 | Torque shutoff closing enabled |
| 9 | External fault generated by FaultExt or external control system fault CSF with set Feature.Bit18 Activating error state for external process control error CSF (Page 162) |
| 10 | Vibration successfully completed – Reset required |
| 11 | Display motor protection (Trip status, Trip.ST ≠ 16#FF) |
| 12 | 1 = Input parameter FbkClose is interconnected |
| 13 | 1 = Input parameter FbkClosing is interconnected |
| 14 | 1 = Input parameter FbkOpen is interconnected |
| 15 | 1 = Input parameter FbkOpening is interconnected |
| 16 | 1 = Input parameter TorOpen is interconnected |
| 17 | 1 = Input parameter TorClose is interconnected |
| 18 | SimLiOp.Value |
| 19 | 1 = Enable for rapid stop (Feature.Bit14 Enabling rapid stop via faceplate (Page 179)) |
| 20 | 1 = Input parameter OpenChnST is interconnected |
| 21 | 1 = Input parameter CloseChnST is interconnected |
| 22 | Not used |
| 23 | Command for rapid stop |
| 24 | "Open"/"Stop" command |
| 25 | "Close"/"Stop" command |
| 26 | Show automatic preview in the standard view |
| 27 | Internal error |
| 28 | GrpErr.Value |
| 29 | RdyToStart.Value |
| 30 | Auxiliary value 1 visible |
| 31 | Auxiliary value 2 visible |

### Status word assignment for Status4 parameter

| Status bit | Parameter |
|---|---|
| 0 | Delay of message ER_AH_Lim |
| 1 | Delay of message ER_AL_Lim |
| 2 | Delay of message RbkWH_Lim |
| 3 | Delay of message RbkWL_Lim |
| 4 | Collection of message delays |
| 5 | ER_AH_En OR Feature.Bit23 |
| 6 | ER_AL_En OR Feature.Bit23 |
| 7 | 1 = Signal manipulated variable difference high limit violation (ER_AH_MsgEn) |
| 8 | 1 = Signal manipulated variable difference low limit violation (ER_AL_MsgEn) |

| Status bit | Parameter |
|---|---|
| 9 | RbkWH_En OR Feature.Bit23 |
| 10 | RbkWL_En OR Feature.Bit23 |
| 11 | 1 = Signal readback value high limit violation (RbkWH_MsgEn) |
| 12 | 1 = Signal readback value low limit violation (RbkWL_MsgEn) |
| 13 | OpenForce.Value |
| 14 | StopForce.Value |
| 15 | CloseForce.Value |
| 16 | 1 = Forced manipulated variable MV_Forced is active |
| 17 | 1 = Tracking value MV_Trk is active |
| 18 | Reset request in automatic mode |
| 19 | Trip.Value |
| 20 | 1 = Vibrate enabled |
| 21 | Internal information for display MV automatic in bar graph |
| 22 | 1= Suppressed messages |
| 23 | Hidden bypass signal in "Permit" |
| 24 | Hidden bypass signal in "Interlock" |
| 25 | Hidden bypass signal in "Protect" |
| 26 | Feature2.Bit2 Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 27 | Feature2.Bit16 Separate interlock for each direction, position, or speed (Page 180) |
| 28 | Current position feedback monitoring time is visible |
| 29 | Current motor monitoring time is visible |
| 30 | Current valve monitoring time is visible |
| 31 | Feature.Bit13 Separate monitoring time for stopping the motor (Page 180) |

### Status word assignment for Status5 parameter

| Status bit | Parameter |
|---|---|
| 0 - 7 | Effective signal 1 to 8 of the message block connected via EventTsIn |
| 8 - 15 | Effective signal 9 to 16 of the message block connected via EventTsIn |
| 16 | AV is not connected |
| 17 | Delay of the message AV_AH_Lim |
| 18 | Delay of the message AV_WH_Lim |
| 19 | Delay of the message AV_TH_Lim |
| 20 | Delay of the message AV_TL_Lim |
| 21 | Delay of the message AV_WL_Lim |
| 22 | Delay of the message AV_AL_Lim |
| 23 | Collection of the message delays |
| 24 | 1 = High limit of the manipulated variable difference is violated (ER_AH_Act.Value) |
| 25 | 1 = Low limit of the manipulated variable difference is violated (ER_AL_Act.Value) |
| 26 | 1 = High limit of the readback value is violated (RbkWH_Act.Value) |
| 27 | 1 = Low limit of the readback value is violated (RbkWL_Act.Value) |

| Status bit | Parameter |
|---|---|
| 28 | 1 = Valve still closes (`Status2.Bit20`) but the motor is not running |
| 29 | 1 = Valve still opens (`Status2.Bit23`) but the motor is not running |
| 30 | `MV_TrkOn.Value` |
| 31 | Not used |

### Status word allocation for `Status6` parameter

| Status bit | Parameter |
|---|---|
| 0 - 15 | Effective signal 1 to 16 of the message block connected via `EventTs2In` |
| 16 | Hidden bypass signal in "Permission Close" |
| 17 | Hidden bypass signal in "Interlock Close" |
| 18 | Hidden bypass signal in "Protection Close" |
| 19 | "Permission Open" button is enabled |
| 20 | "Permission Close" button is enabled |
| 21 | "Interlock Open" button is enabled |
| 22 | "Interlock Close" button is enabled |
| 23 | "Protection Open" button is enabled |
| 24 | "Protection Close" button is enabled |
| 25 | 1 = "Permission Close" is active |
| 26 | 1 = "Interlock Close" is active |
| 27 | 1 = "Protection Close" is active |
| 28 | 1 = "Permit neutral pos." or " Permission stop" button is enabled |
| 29 | 1 = Permission SafePos / Stop is active |
| 30 | Hidden bypass signal in Permission SafePos / Stop |
| 31 | Not used |

### Status word allocation for `Status7` parameter

| Status bit | Parameter |
|---|---|
| 0 | 1 = Open operation disabled |
| 1 | 1 = Close operation disabled |
| 2 | Resynchronization: Valve Open |
| 3 | Resynchronization: Valve Close |
| 4 | Resynchronization: Valve Stop |
| 5 | Resynchronization: Motor Opening |
| 6 | Resynchronization: Motor Closing |
| 7 | Resynchronization: Motor Stopped |
| 8 | Resynchronization: Display preview |
| 9 | Resynchronization: Display small preview symbol in standard view |
| 10 | 1 = Operator MV input in open direction is not allowed |
| 11 | 1 = Operator MV input in close direction is not allowed |

| Status bit | Parameter |
|---|---|
| 12 | Resynchronization preview view for Valve/Motor open |
| 13 | Resynchronization preview view for Valve/Motor close |
| 14 | Resynchronization preview view for Valve/Motor Stop |
| 15 - 31 | Not used |

### See also

VlvPosL functions (Page 1884)

VlvPosL messaging (Page 1910)

VlvPosL block diagram (Page 1928)

VlvPosL error handling (Page 1908)

VlvPosL modes (Page 1882)

## 8.14.2 VlvPosL modes

### VlvPosL operating modes

The block supports all standard operating modes:

- Local mode (Page 86)
- Automatic mode (Page 82)
- Manual mode (Page 82)
- Out of service (Page 77)

The next section provides additional block-specific information relating to the general descriptions.

### "Local mode"

You can find general information on "Local mode", switching modes, and bumpless switchover in the section Local mode (Page 86).

When you put a block into "Local mode", it is controlled either by "Local" signals or by feedback signals. With `LocalSetting` = 1 or 3, you can control the motor valve with the following "Local" signals:

- "Open" (`OpenLocal` = 1)
- "Close" (`CloseLocal` = 1)
- "Stop" (`StopLocal` = 1)

The feedback signals `FbkOpening`, `FbkClosing`, `FbkOpen`, `FbkClose`, and the analog readback `Rbk` will be monitored.

With `LocalSetting` = 2 or 4, the control outputs `Open`, `Close` and `StopCtrl` and the internal status `Opened`/`Closed`/`Stop` of the block will be tracked.

In the case where motor feedback signals are available (`Feature.Bit12` = 0), the control outputs `Open`, `Close` and `StopCtrl` will be tracked from the motor feedback signals `FbkOpening` and `FbkClosing`. The valve feedback signals `FbkOpen`, `FbkClose`, and the analog readback `Rbk` will be monitored.

In the case where no motor feedback signals are available (`Feature.Bit12` = 1), the internal status `Opened`/`Closed`/`Stop` of the block will be tracked from the valve feedback signals `FbkOpen`, `FbkClose`, and the analog readback `Rbk`. It will only be monitored if the analog readback and valve feedback signals are not suitable together.

If no position can be identified, the last valid position is assumed.

In local mode position control is switched off and manipulated value `MV_Out` is tracked to `MV_Scale.High`, `MV_Scale.Low`, `RbkOut`.

### "Automatic mode"

You can find general information on "Automatic mode", switching modes and bumpless switchover in the section Manual and automatic mode for motors, valves and dosers (Page 82).

You can control the motor valve in "Automatic mode" with analog position feedback:

- "Open" (`OpenAut` = 1)
- "Close" (`CloseAut` = 1)
- "Stop" (`StopAut` = 1)

Position control is terminated with "Open", "Close", or "Stop". It is enabled again with:

- "Activate position control" (`PosOnAut` = 1)

For position control, the motor valve is controlled via the `Open` and `Close` outputs such that the manipulated variable `MV` and position feedback `Rbk` come into agreement.

### "Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the section Manual and automatic mode for motors, valves and dosers (Page 82).

You can control the motor valve in "Manual mode" with analog position feedback:

- "Open" (`OpenMan` = 1)
- "Close" (`CloseMan` = 1)
- "Stop" (`StopMan` = 1)

Position control is terminated with "Open", "Close", or "Stop". It is enabled again with:

- "Activate position control" (`PosOnMan` = 1)

For position control, the motor valve is controlled via the `Open` and `Close` outputs such that the manual value `Man` and position feedback `Rbk` come into agreement.

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature2.Bit1` =0 and `Feature2.Bit1` =1 is as shown below:

| Output Parameters | **`Feature2.Bit1` =0** | **`Feature2.Bit1` =1** |
|---|---|---|
| FbkOpenOut.ST | 16#80 | 16#60 |
| FbkCloseOut.ST | 16#80 | 16#60 |
| FbkOpngOut.ST | 16#80 | 16#60 |
| FbkClsgOut.ST | 16#80 | 16#60 |
| Opened.ST | 16#80 | 16#60 |
| Closed.ST | 16#80 | 16#60 |
| Opening.ST | 16#80 | 16#60 |
| Closing.ST | 16#80 | 16#60 |
| Stop.ST | 16#80 | 16#60 |
| RbkOut.ST | 16#XX | 16#60 |
| RbkWH_Act.ST | 16#XX | 16#60 |
| RbkWL_Act.ST | 16#XX | 16#60 |
| PosReached.ST | 16#80 | 16#60 |
| ER.ST | 16#80 | 16#60 |
| ER_AH_Act.ST | 16#XX | 16#60 |
| ER_AL_Act.ST | 16#XX | 16#60 |
| AV_Out.ST | 16#80 | 16#60 |

### See also

VlvPosL block diagram (Page 1928)
VlvPosL I/Os (Page 1912)
VlvPosL error handling (Page 1908)
VlvPosL functions (Page 1884)
VlvPosL messaging (Page 1910)
Description of VlvPosL (Page 1876)

## 8.14.3 VlvPosL functions

### Functions of VlvPosL

The functions for this block are listed below.

### Motor valve control

The motor valve is controlled via the Open and Close outputs or via the pulse outputs P_Open, P_Close, and P_Stop. The outputs are set by either of the following methods:

- Directly via the corresponding inputs:
  - OpenMan, CloseMan, and StopMan in manual mode
  - OpenAut, CloseAut, and StopAut in automatic mode
  - OpenLocal, CloseLocal, and StopLocal in local mode
  - OpenForce, CloseForce, and StopForce for the force operating states function. The position control is switched off. The analog control output MV_Out is tracked:
    OpenXxx: MV_Scale.High
    CloseXxx: MV_Scale.Low
    StopXxx: Rbk_Out
    with Xxx: Man,Aut,Local,Force
- Calculated by a position controller which controls the position feedback signal Rbk to a specified manipulated variable MV_Out. Depending of mode and function MV_Out is calculated by the following inputs:
  - Man in manual mode
  - MV in automatic mode
  - MV_Trk when „Tracking function (MV_TrkOn =1)“ is active.
  - MV_Force when „Forcing function for analog value (MV_ForOn =1)“ is active.

Position control is switched on by:

- PosOnMan in manual mode
- PosOnAut in automatic mode
- MV_ForOn for the force operating states functions.

Position control is switched off by:

- direct control via the command inputs OpenX, CloseX, StopX (see above)
- after a switch to Local Mode
- after a switch to Out of Service mode
- after a monitoring error of the feedbacks FbkOpen,FbkClose,Rbk

In the local mode, the manipulated variable is always tracked to the position feedback.

For direct control via the Open, Close, Stop inputs above, position control is terminated and must be reactivated via the PosOn inputs.

### Motor valve control with position control

When controlling with enabled position control PosAct = 1, the motor valve is moved to the desired valve position using the specified analog manipulated variable.

Depending on the mode and function, the effective analog manipulated variable `MV_Out` is specified by:

| FaultExt<br>CLS<br>+FB18<br>Trip<br>RapidStp | MV_ForOn | Permit Intlock Protect ...Cls | ManAct | MV_TrkOn | MV_Out<br>Open, Close are calculated from position controller | Limits | Comment |
|---|---|---|---|---|---|---|---|
| 1 | - | - | - | - | `FaultExt,CLS+FB8:`<br>Depending on rest position:<br>`SafePos = 0:`<br>`MV_Out = MV_LoLim`<br>`SafePos = 1:`<br>`MV_Out = MV_HiLim`<br>`SafePos = 2:`<br>`MV_Out = RbkOut`<br>`Trip, RapidStp:`<br>`MV_Out = RbkOut` | `MV_HiLim`<br>`MV_LoLim` | External fault, Motor protection, Rapid stop |
| 0 | 1 | - | - | - | `MV_Out = MV_Forced` | none | Forced tracking |
| 0 | 0 | 1 | - | - | Interlock is independent of direction:<br>Depending on rest position:<br>`SafePos = 0:`<br>`MV_Out = MV_LoLim`<br>`SafePos = 1:`<br>`MV_Out = MV_HiLim`<br>`SafePos = 2:`<br>`MV_Out = RbkOut`<br>Interlock is dependent of direction:<br>Dependent on `SafePos`, drive direction and interlock direction `MV_Out` is tracked to `MV_HiLim`, `MV_LoLim`, `RbkOut`. | `MV_HiLim`<br>`MV_LoLim` | Interlocks |
| 0 | 0 | 0 | 1 | - | `Man` | `MV_HiLim`<br>`MV_LoLim` | Manual mode |
| 0 | 0 | 0 | 0<br>`AutAct` = 1 | 1 | `MV_Trk` | `MV_HiLim`<br>`MV_LoLim` | Tracking |
| 0 | 0 | 0 | `0`<br>`AutAct` = 1 | 0 | `MV` | `MV_HiLim`<br>`MV_LoLim` | Automatic mode |

Functions like sealing, torque monitoring or vibrations are not included in the table.

### Motor valve control without position control

The commands for the controlling of motor valve are inputed over `OpenX, CloseX, StopX` (see above). The analog manipulated value output `MV_Out` is tracking:

| FaultExt<br>CLS<br>+FB18<br>Trip<br>Rap-<br>idStp | OpenForce<br>CloseForce<br>StopForce | Permit Intlock<br>Protect ...Cls | OpenX<br>CloseX<br>StopX | Open, Close<br>MV_Out is tracking | Limits, Range | Comment |
|---|---|---|---|---|---|---|
| 1 | - | - | - | `FaultExt,CLS+FB8`: Depending on rest position: `SafePos = 0: Close = 1, MV_Out = MV_Scale.Low`<br>When end position is reached: `Close =0`<br>`SafePos = 1: Open = 1, MV_Out = MV_Scale.High`<br>When end position is reached: `Open =0`<br>`SafePos = 2: Close, Open = 0, MV_Out = RbkOut`<br>`Trip, RapidStp: Close, Open = 0, MV_Out = RbkOut` | `MV_Scale.High`<br>`MV_Scale.Low` | External fault, Motor protection, Rapid stop |
| 0 | 1 | - | - | `CloseForce=1: Close = 1, MV_Out = MV_Scale.Low`<br>When end position is reached: `Close =0`<br>`OpenForce = 1: Open = 1, MV_Out = MV_Scale.High`<br>When end position is reached: `Open = 0`<br>`StopForce = 1: Close,Open = 0, MV_Out = RbkOut` | `MV_Scale.High`<br>`MV_Scale.Low` | Forcing position |

| FaultExt<br>CLS<br>+FB18<br>Trip<br>Rap-<br>idStp | OpenForce<br>CloseForce<br>StopForce | Permit Intlock<br>Protect ...Cls | OpenX<br>CloseX<br>StopX | Open, Close<br>MV_Out is tracking | Limits, Range | Comment |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | - | Interlock is independent of direction:<br>Depending on rest position: `SafePos = 0: Close = 1, MV_Out = MV_Scale.Low`<br>When end position is reached: `Close = 0 SafePos = 1: Open = 1, MV_Out = MV_Scale.High`<br>When end position is reached: `Open = 0 SafePos = 2: Close, Open = 0, MV_Out = RbkOut`<br>Interlock is dependent of direction:<br>Dependent on `SafePos`, drive direction and interlock direction Open, Close are set and `MV_Out` is tracked to `MV_Scale.High, MV_Scale.Low, RbkOut.` | `MV_Scale.High`<br>`MV_Scale.Low` | Interlocks |
| 0 | 0 | 0 | 1 | `CloseX = 1: Close = 1, MV_Out = MV_Scale.Low`<br>When end position is reached: `Close = 0 OpenX = 1: Open = 1, MV_Out = MV_Scale.High`<br>When end position is reached: `Open =0 StopX = 1:Close, Open = 0, MV_Out = RbkOut`<br>After a stop control, the manipulated variable is tracked for the time from `V_MonTiStatic` to the position feedback (`MV_Out = RbkOut`). The manipulated variable is then held and can be monitored | `MV_Scale.High`<br>`MV_Scale.Low` | Manual mode<br>Automatic mode<br>Local mode<br>X: Man<br>Aut<br>Local |

Functions like sealing, torque monitoring or vibrations are not included in the table.

If the motor valve is in "Out Of Service" mode (OOS), the analog position control is switched off. The controls Open and Close are set depending on the parameters SafePos, Rbk, FbkOpen, and FbkClose. Accordingly, the output MV_Out is tracked to MV_HiLim, MV_LoLim or RbkOut. The control outputs Open and Close are used to put the valve into its target position. If the target position is lost, then the control outputs are set again.

### Limiting the manipulated variable MV_Out

If position control is on (PosAct =1) the manipulated variable is limited by MV_HiLim and MV_LoLim and output at MV_Out. Reaching the limit is displayed at MV_HiAct =1 for the high limit and MV_LoAct =1 for the low limit.

**Note**

If position control is off (PosAct =0) or forcing is active over OpenForce, CloseForce, StopForce or MV_ForOn (independent of position control) the MV_Out is not limited and the outputs MV_HiAct, MV_LoAct are set to false.

### Deadband for position control

The manipulated variable difference MV_Out – RbkOut is controlled by a deadband width (DeadBand) and displayed at the ER output.

Depending on the parameter DeadBand and the difference (MV_Out – RbkOut) the following cases can occur:

- **(MV_Out – RbkOut) > DeadBand:**
  ER = (MV_Out – RbkOut) – DeadBand
- **(MV_Out – RbkOut) < (–1* DeadBand):**
  ER = (MV_Out – RbkOut) + DeadBand
- **Otherwise:**
  ER = 0.0

The internal position controller is set with the following parameters:

- Gain: Controller gain
- MotorTime: Motor runtime from closed to fully open valve
- TmLag: Delay time caused, for example, by lag of the motor valve when switching on and off.

The pulse output is formed by a three-step element with hysteresis and set with the following parameters:

- PulseTime: Minimum pulse length of the outputs Open or Close
- BreakTime: Mminimum pulse break between two pulse outputs of Open or Close

**Note**

PulseTime and BreakTime are limited to SampleTime. If they are smaller parameterized than SampleTime they will be written back to the input with the value of SampleTime.

PulseTime and BreakTime values which are too large, can reduce the precision of position control.

The effective trip threshold and response threshold of the three-step element is displayed at the `ThreshOff` and `ThreshOn` output parameters.

The `PosReached` output shows whether the specified position has been reached. This is the case when:

- Intermediate position:
  `PosRbkHys` >= ABS (`MV_Out` - `RbkOut`)
- End position open:
  `MV_Out` >= `PosDiOpen` AND `RbkOut` >= `PosDiOpen` AND `FbkOpenOut` AND NOT `FbkCloseOut`
- End position closed:
  `MV_Out` <= `PosDiClose` AND `RbkOut` <= `PosDiClose` AND `FbkCloseOut` AND NOT `FbkOpenOut`

**Note**

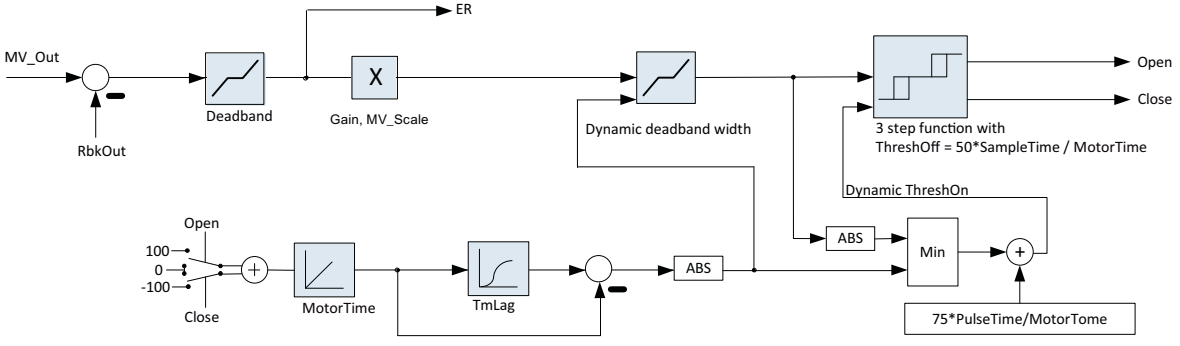
For the end positions "`PosDiOpen`" and "`PosDiClose`", the hysteresis and deadband also apply.

If the position is reached, the control and feedback signals of the motor are reset with the following values:

- `Open` = 0
- `Close` = 0
- `FbkOpngOut` = 0
- `FbkClsgOut` = 0

### Position control for control valves with delay behavior

The purpose of the position control is to move the valve to a specified control valve position by switching on and off as few times as possible at the `Open` and `Close` outputs.

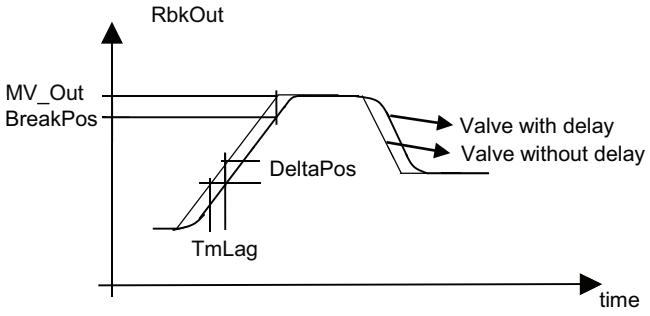



The control difference between the output `MV_Out` and the position feedback signal `RbkOut` can be amplified at the `Gain` parameter.

Small control differences are suppressed at the dead band using the `DeadBand` parameter. As a rule, the preset values `Gain` = 1 and `DeadBand` = 0 can be retained.

At second dead zone, the dead zone width is dynamically formed as a function of `TmLag`. `TmLag` represents a delay time that can occur when the control valve moves.

If `TmLag` =0, the dynamic is switched off and the result is a dead zone width of zero.




Derived from the binary outputs `Open`, `Close`, and the time constant `MotorTime`, `TmLag` (the dynamic dead band width) is calculated as a difference between a delayed and an undelayed motor valve. This ensures that the pulses are switched off via `Open` and `Close` on time to reach the specified valve position without overshoot.

For example: The calculation of `Open` and `Close` to avoid overshoot is done as follows:

`DeltaPos` = 100% * `TmLag` / `MotorTime`

- Case A: `MV_Out` > `RbkOut`: `BreakPos` = `MV_Out` – `DeltaPos`
- Case B: `MV_Out` < `RbkOut`: `BreakPos` = `MV_Out` + `DeltaPos`
  - `DeltaPos` is the distance to the target position.
  - `BreakPos` is the calculated feedback position before reaching the target position where the control outputs `Open` and `Close` switch off in time in order to avoid overshoot.

A three step function with hysteresis is used to calculate the binary control outputs `Open` and `Close`, from the manipulated variable difference `ER`. The number of pulses required to reach a preset valve position is further reduced by an adaptive response threshold of the three step function. The dynamic behavior is also switched off with `TmLag` = 0. The position control is switched off after a monitoring error.

## Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

## Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the parameter `OS_Perm`:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can switch to "Automatic mode" |
| 1 | 1 = Operator can switch to "Manual mode" |
| 2 | 1 = Operator can switch to "Local mode" |
| 3 | 1 = Operator can switch to "Out of service mode" |

| Bit | Function |
|---|---|
| 4 | 1 = Operator can stop the motor |
| 5 | 1 = Operator can open the valve |
| 6 | 1 = Operator can close the valve |
| 7 | 1 = Operator can trigger rapid stop for motors |
| 8 | 1 = Operator can activate position control |
| 9 | 1 = Operator can reset the valve |
| 10 | 1 = Operator can change the manual value `Man` |
| 11 | 1 = Operator can change the simulation value `SimRbk` |
| 12 | 1 = Operator can define the valve monitoring time for control |
| 13 | 1 = Operator can define the valve monitoring time for end position |
| 14 | 1 = Operator can activate the function valve monitoring time (bits 12 and 13) |
| 15 | 1 = Operator can define the motor monitoring time for control start or stop |
| 16 | 1 = Operator can define the monitoring time for control stop |
| 17 | 1 = Operator can define the motor monitoring time for status |
| 18 | 1 = Operator can activate the function motor monitoring time (bits 15,16, and 17) |
| 19 | 1 = Operator can define the readback monitoring time |
| 20 | 1 = Operator can activate the function readback monitoring time (bit 19) |
| 21 | Not used |
| 22 | 1 = Operator can change proportional gain of the position control |
| 23 | Not used |
| 24 | 1 = Operator can set the delay time of the motor valve |
| 25 | 1 = Operator can set the deadband width of the manipulated variable difference |
| 26 | 1 = Operator can set the motor run time |
| 27 | 1 = Operator can set the minimum pulse duration |
| 28 | 1 = Operator can set the minimum break duration |
| 29 | Not used |
| 30 | 1 = Operator can activate the "Simulation" function |
| 31 | 1 = Operator can activate the "Release for maintenance" function |

The block has the following permissions for the parameter `OS1Perm`:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can change the high operating limit of the manipulated variable `MV_HiLim` |
| 1 | 1 = Operator can change the low operating limit of the manipulated variable `MV_LoLim` |
| 2 | Not used |
| 3 | 1 = Operator can enable/disable messages via `ER_AH_MsgEn` |
| 4 | 1 = Operator can enable/disable messages via `ER_AL_MsgEn` |
| 5 | 1 = Operator can enable/disable messages via `RbkWH_MsgEn` |
| 6 | 1 = Operator can enable/disable messages via `RbkWL_MsgEn` |
| 7 | 1 = Operator can change the limit (manipulated variable difference) for the high alarm `ER_AH_Lim` |
| 8 | 1 = Operator can change the hysteresis (manipulated variable difference) `ER_Hyst` |
| 9 | 1 = Operator can change the limit (manipulated variable difference) for the low alarm `ER_AL_Lim` |

| Bit | Function |
|---|---|
| 10 | 1 = Operator can change the limit (position feedback) for the high warning RbkWH_Lim |
| 11 | 1 = Operator can change the hysteresis (position feedback) RbkHyst |
| 12 | 1 = Operator can change the hysteresis (position feedback) for the low warning RbkWL_Lim |
| 13 | Not used |
| 14 | 1 = Operator can change the simulation value SimAV |
| 15 | 1 = Operator can change the limit (AV) for the high alarm |
| 16 | 1 = Operator can change the limit (AV) for the high warning |
| 17 | 1 = Operator can change the limit (AV) for the high tolerance |
| 18 | 1 = Operator can change the limit (AV) for the hysteresis |
| 19 | 1 = Operator can change the limit (AV) for the low alarm |
| 20 | 1 = Operator can change the limit (AV) for the low warning |
| 21 | 1 = Operator can change the limit (AV) for the low tolerance |
| 22 | 1 = Operator can enable / disable messages via AV_AH_MsgEn |
| 23 | 1 = Operator can enable / disable messages via AV_WH_MsgEn |
| 24 | 1 = Operator can enable / disable messages via AV_TH_MsgEn |
| 25 | 1 = Operator can enable / disable messages via AV_TL_MsgEn |
| 26 | 1 = Operator can enable / disable messages via AV_WL_MsgEn |
| 27 | 1 = Operator can enable / disable messages via AV_AL_MsgEn |
| 28 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in OS_Perm, OS1Perm as a parameter, you have to reset the corresponding OS_Perm, OS1Perm bit.

### Limit monitoring for additional analog value

This block provides the standard function Limit monitoring of an additional analog value (Page 98).

### Limit monitoring with hysteresis

This block provides the standard function Limit monitoring with hysteresis (Page 104). This is done using the input parameter AV_Hyst.

### Providing limits at the output

For further connections to other blocks the following input parameters will also be displayed on the output:

```
ER_AH_LimOut := ER_AH_Lim;
ER_AL_LimOut := ER_AL_Lim;
```

### Suppress messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Interlocks

This block provides the following interlocks:

- `Feature2.Bit16` = 0
  - Activation enable ("Permission")
  - Interlock without reset ("Interlock")
  - Interlock with reset ("Protection")
  - Permission for safe position (“Permit neutral pos.” or “Permission stop”)
- `Feature2.Bit16` = 1
  - Activation enable Open ("Permission Open")
  - Activation enable Close ("Permission Close")
  - Interlock open without reset ("Interlock Open")
  - Interlock close without reset ("Interlock Close")
  - Interlock open with reset ("Protection Open")
  - Interlock close with reset ("Protection Close")

For more on this, see the sections Interlocks (Page 107) and Influence of the signal status on the interlock (Page 112).

When an interlock is pending and position control is enabled, the block remains in position control. The following manipulated variables are controlled based on the configured neutral position for an interlock with/without reset:

| `SafePos` | Manipulated variable |
|---|---|
| 0: Closed | `MV_Out` = `MV_LoLim` |
| 1: Opened | `MV_Out` = `MV_HiLim` |
| 2: Stopped | `MV_Out` = `RbkOut` |

**Note**

The neutral position cannot be reached if the limit `MV_HiLim` is not configured greater than or equal to `PosDiOpen` or `MV_LoLim` is not configured less than or equal to `PosDiClose`.

With VlvPosL in position control, the neutral position is reached when:

| `SafePos` | Neutral position is reached when |
|---|---|
| 0: Closed | `PosReached` = 1, `FBkOpenOut` = 0, `FBkCloseOut` = 1 |
| 1: Opened | `PosReached` = 1, `FBkOpenOut` = 1, `FBkCloseOut` = 0 |
| 2: Stopped | `PosReached` = 1, independent of `FBkOpenOut`, `FBkCloseOut` |

If VlvPosL is in the neutral position, it can only leave this position when an activation enable is set.

### Motor protection function

This block provides the standard function Motor protection function (Page 106).

When motor protection is pending and position control is enabled, the block remains in position control. The manipulated variable is tracked to the feedback value (`MV_Out` = `RbkOut`) and with this, the motor is in the stop state.

### Rapid stop for motors

This block provides the standard function Rapid stop for motors (Page 115).

When rapid stop is pending and position control is enabled, the block remains in position control. The manipulated variable is tracked to the feedback value (`MV_Out` = `RbkOut`) and with this, the motor is in the stop state.

### Torque monitoring

This block provides torque monitoring.

The signals of the torque monitoring switches are interconnected to the input parameters `TorqOpen` to open the motor valve and `TorqClose` to close the motor valve.

- `TorqOpen` / `TorqClose` (`Value` =1 and `ST`<>16#00, 16#28): Torque in good state
- `TorqOpen` (`Value` =0 or `ST`=16#00, 16#28): Torque increased in direction Up
- `TorqClose` (`Value` =0 or `ST`=16#00, 16#28): Torque increased in direction Closed

Using `Feature2.Bit6` (Vibration), you can define how to proceed in automatic mode when torque is active. If `Feature.Bit8` is set, the parameter `TorqClose` is also considered for the function "Sealing the valve".

**Vibration is deactivated (`Feature2.Bit6` = 0)**

If one of the two torques `TorqOpen` or `TorqClose` is activated, a running motor is stopped in the corresponding direction and the position control is switched off. You can still move the valve in the opposite direction. If both the torques are activated, the motor valve can no longer be moved.

An active torque is displayed in the standard view as "Torque active" in the display area for block states.

**Vibration is activated (`Feature2.Bit6` =1) and VlvPosL is in automatic mode**

Depending on whether the position control is switched on, the attempts of vibration refer to the parameter `VibrWidth` in seconds or the parameter `VibrPerc` in percent.

- **Vibration with position control switched off (`PosAct` =0)**
  - If a torque occurs to one of the end positions during a control, the motor runs in the opposite direction for `VibrWidth` seconds.
  - If the torque normalizes again after `VibrWidth` seconds, the motor is controlled in the originally desired direction. When the torque comes active again, the motor runs in the opposite direction again for `VibrWidth` seconds. This procedure is repeated for a maximum of `VibrNo` times.
- **Vibration with position control switched on (`PosAct` =1)**
  - If a torque occurs to a new valve position `MV` during control, the motor runs back to the position `MV_Out` = `RbkOut` - (`MV_HiLim` - `MV_LoLim`) / 100.0 * `VibrPerc` in the opposite direction.
  - If the torque has normalized again after reaching to this position, the control takes place in the originally desired position `MV`. When the torque is switched off again, the motor valve runs back by `VibrPerc` percent. This procedure is repeated a maximum of `VibrNo` times.

If the function vibration is activated, "Vibrate enabled" is displayed in the display area of the standard view.

After `VibrNo` attempts of reverse moving and control again to the original position without reaching this position because of an active torque, the motor stops and position control is switched off. You must reset the block so that vibration can be reactivated.

If the torque does not normalize during the reverse moving, the motor stops and the position control is switched off.

**Note**

The function vibration tries to release a blockage by repeatedly moving the motor valve back and forth against it. This is only suitable for certain types of blockages, such as clearing branches or chunks of wood from weir valves.

### Sealing the valve

This function is enabled using `Feature.Bit8` Sealing the valve (Page 187). The seal valve function combines the query of the end position "Closed" via the input parameter `FbkClose` and `Rbk` with the limit violation of the configured torque via the input parameter `TorqClose`. This ensures that the valve is completely closed.

The valve is only considered completely closed when the feedback for the end position "Close" has been received (0->1), the position feedback `Rbk` is smaller than or equal to `PosDiClose`, and the torque shutdown for "Closed" is enabled. The torque shutoff should not come before the

feedback in this case. The FbkCloseOut output shows whether the valve is sealed tight with the following conditions:

- FbkCloseOut:= FbkClose.Value AND RbkOut.Value <= PosDiClose has come in (0->1)
- "Torque shutoff closed is enabled"
- "Torque shutoff closed is enabled" did not come before FbkCloseOut (0->1)

"Torque shutoff closed enabled" means TorqClose = 0 or the signal status is 16#00 or 16#28.

When "Torque shutoff closed is enabled" comes before the end position feedback closed, this is displayed in the faceplate in the standard view and the motor stops. Opening the valve is still possible.

**Note**

The command text for Close in the standard view and preview can be changed to Seal in the CFC at the CloseMan parameter in Text 1. See section Labeling of buttons and text (Page 235).

### Disable interlocks

This block provides the standard function Disabling interlocks (Page 111).

### Resetting the block in case of interlocks

This block provides the standard function Resetting the block in case of interlocks or errors (Page 47).

### External fault (FaultExt), External control system fault (CSF)

This block provides the possibility to pass an external fault via the FaultExt parameter or an external control system fault via the CSF parameter. See VlvPosL error handling (Page 1908).

When an external fault or control system fault is pending (Feature.Bit18 = 1) and position control is enabled, the block remains in position control. The following manipulated variable is set based on the configured neutral position:

| SafePos | Manipulated variable |
|---|---|
| 0: Closed | MV_Out = MV_LoLim |
| 1: Opened | MV_Out = MV_HiLim |
| 2: Stopped | MV_Out = RbkOut |

**Note**

The neutral position cannot be reached if the limit MV_HiLim is not configured greater than or equal to PosDiOpen or MV_LoLim is not configured less than or equal to PosDiClose.

### Group error

This block provides the standard function, Outputting group errors (Page 130).

The following parameters are taken into consideration when forming the group error:

- `CSF`
- `Trip`
- `V_MonDynErr`
- `V_MonStaErr`
- `M_MonDynErr`
- `M_MonStaErr`
- `RbkMonDynErr`
- `RbkMonStaErr`
- `FaultExt`

### Outputting a signal for start readiness

This block provides the standard function, Outputting a signal for start readiness (Page 59).

### Forming the group status for interlocks

This block provides the standard function, Forming the group status for interlock information (Page 113).

### Control loop is open (`CascaCut`)

The `CascaCut` output indicates whether the control loop within this block is open. In the open control loop, the master controller must be set to tracking mode to avoid until windup behavior. If you use this block in a cascade structure, then connect the output `CascaCut` to the input `MV_TrkOn` of the master controller.

`CascaCut` is active in the following cases:

- Block is not in automatic mode
- Force function is active
- Motor protection is active
- Interlock is active
- Tracking function is active
- Position control is off
- Monitoring error is active

### Forming the signal status for blocks

This block provides the standard function, Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status for the block is formed from the following parameters:

- FbkClsgOut.ST
- FbkOpngOut.ST
- FbkOpenOut.ST
- FbkCloseOut.ST
- RbkOut.ST
- LocalLi.ST
- OpenLocal.ST
- StopLocal.ST
- CloseLocal.ST
- Trip.ST
- TorqOpen.ST
- TorqClose.ST
- AV_Out.ST
- OpenChn.ST
- CloseChn.ST
- OpenAut.ST (only if Feature2.Bit10 = 1)
- CloseAut.ST (only if Feature2.Bit10 = 1)
- StopAut.ST (only if Feature2.Bit10 = 1)
- MV.ST (only if Feature2.Bit10 = 1)
- PosOnAut.ST (only if Feature2.Bit10 = 1)
- MV_Out.ST

### Considering bad quality of automatic commands or external values

If the Feature2 bit Considering bad quality of automatic commands or external values (Page 200) is set to 1 and one of the following parameters has bad signal status (16#00 or 16#28), the block goes to the neutral position in the "Automatic" mode:

- OpenAut.ST
- CloseAut.ST
- StopAut.ST
- MV.ST
- PosOnAut.ST

## Forcing operating modes

This block provides the standard function, Forcing operating modes (Page 45). The inputs `OpenForce`, `CloseForce`, and `StopForce` force the block to open, close, or stop, respectively. When any of these inputs is set, then the activated position control `PosAct` is terminated and you must separately reactivate it.

With `MV_ForOn` = 1, position control is enabled and set to the manipulated variable `MV_Forced`. The inputs `OpenForce`, `CloseForce` and `StopForce` have higher priority over `MV_ForOn`.

## Feedback monitoring

This block provides the standard function, Monitoring the feedbacks (Page 104).

The `FbkOpen` and `FbkClose` feedback is monitored for the valve; the `FbKOpening` and `FbkClosing` feedback is monitored for the motor.

The monitoring of the feedback for the valve will not be active if it was stopped during opening or closing.

### Monitoring valve feedback

The monitoring of valve feedback is set using the parameter `V_Monitor`.

Startup characteristics are monitored by setting parameter `V_MonTiDynamic`. The parameter `V_MonTiStatic` monitors compliance with the position.

Feedback errors are displayed at the corresponding parameters `V_MonDynErr` and/or `V_MonStaErr`.

**Note**

After the motor valve stops in the intermediate position or end position or after a startup without control (`Open`, `Close` = 0)" or a stop after a monitoring error, no monitoring of the feedback signals `FbkOpen` and `FbkClose` takes place during the `V_MonTiStatic` time. Changes to `FbkOpen` and `FbkClose` are applied. This means that the feedback is monitored again, also in stop state.

**Note**

When the "Seal valve" function is enabled via `Feature.Bit8`, the torque shutoff for closing `TorqClose` is also evaluated (see section Sealing the valve (Page 187)).

### Monitoring the motor feedback

The monitoring of motor feedback is set using the parameter `M_Monitor`.

Startup characteristics are monitored by setting parameter`M_MonTiDynamic`. The parameter `M_MonTiStatic` monitors compliance with the position.

Feedback errors are displayed at the corresponding parameters `M_MonDynErr` and/or `M_MonStaErr`.

### Monitoring valve feedback

The monitoring of valve feedback is set using the V_Monitor parameter.

Startup characteristics are monitored by setting the V_MonTiDynamic parameter. When starting the motor valve, the current position feedback Rbk is taken into consideration and is included in the calculation of the monitoring time as follows:

Monitoring of the position feedback is switched on (RbkMonitor =1):

- Monitoring time = V_MonTiDynamic – MotorTime + Abs(MV_Out - RbkOut) * MotorTime/(MV_Scale.High-MV_Scale.Low)

**Note**

The dynamic monitoring time V_MonTiDynamic should be greater than motor actuating time MotorTime.

Monitoring of the position feedback is switched off (RbkMonitor =0):

- Open or close motor valve: Monitoring time = V_MonTiDynamic

Monitoring of the position feedback Rbk with activated position control can be set separately using RbkMonitor, RbkMonTi.

The V_MonTiStatic parameter sets the compliance with the valve position.

Feedback errors are displayed at the corresponding parameters V_MonDynErr or V_MonStaErr.

The valve travel range is divided into three ranges for monitoring:

- Valve open: RbkOut ≥ PosDiOpen
- Valve in intermediate position: PosDiClose < RbkOut < PosDiOpen
- Valve closed: RbkOut ≤ PosDiClose

To monitor the end positions, the FbkOpenOut and FbkCloseOut outputs are formed as follows:

When the motor valve reaches the "Open" position, the output is FbkOpenOut = 1:

- With a binary limit switch for the "Open" position (NoFbkOpen = 0):
  The valve is considered open if FbkOpenOut is set. FbkOpenOut is set if RbkOut ≥ PosDiOpen and FbkOpen = 1.
- Without binary end switch for the position "Open" (NoFbkOpen = 1):
  The valve is considered open if FbkOpenOut is set. FbkOpenOut is set if RbkOut ≥ PosDiOpen.

When the motor valve reaches the "Closed" position, the output is FbkCloseOut = 1:

- With a binary limit switch for the "Closed" position (NoFbkClose = 0):
  The valve is considered closed if FbkCloseOut is set. FbkCloseOut is set if RbkOut ≤ PosDiClose and FbkClose = 1.
- Without binary end switch for the position "Closed" (NoFbkClose = 1):
  The valve is considered closed if FbkCloseOut is set. FbkCloseOut is set if RbkOut ≤ PosDiClose.

**Note**

After the motor valve stops in the intermediate position or end position or after a CPU startup without control (`Open`, `Close` = 0), no monitoring of the feedback signals `FbkOpen`, `FbkClose` and `Rbk` takes place during the time `V_MonTiStatic`. Changes to `FbkOpen`, `FbkClose` and `Rbk` are applied to `FbkOpenOut` and `FbkCloseOut`. After the time `V_MonTiStatic`, the feedbacks are monitored again, also in the stop state.

**Note**

When the "Seal valve" function is enabled by means of `Feature.Bit8`, the torque shutoff for the closing `TorqClose` is also evaluated (see section Sealing the valve (Page 187)).

### Monitoring the motor feedback

The monitoring of motor feedback is set using the `M_Monitor` parameter.

Startup characteristics are monitored by setting the `M_MonTiDynamic` parameter; the `M_MonTiStatic` parameter monitors maintenance of the position.

Feedback errors are displayed at the corresponding parameters `M_MonDynErr` or `M_MonStaErr`.

### Monitoring of the position feedback Rbk

The monitoring of position feedback `Rbk` is set using the `RbkMonitor` parameter.

The monitoring of the startup characteristics is set at the `RbkMonTi` parameter. The effective monitoring time is calculated as follows:

- Monitoring time = ABS($MV_Out_n$ - $MV_Out_{n-1}$) * `MotorTime`/(`MV_Scale.High`–`MV_Scale.Low`) + `RbkMonTi`
  $MV_Out_n$: Current manipulated variable to be reached
  $MV_Out_{n-1}$: Last manipulated variable reached

The valve position is reached when the following condition is met:
`PosRbkHys` >= ABS(`MV_Out` - `RbkOut.Value`)

Feedback errors are displayed at the corresponding parameters `RbkMonDynErr` or `RbkMonStaErr`.

### Release for maintenance

This block provides the standard function, Release for maintenance (Page 70).

### Specify warning times for control functions

This block provides the standard function, Specifying warning times for control functions at motors and valves (Page 56).

You can generate warning signals when, for example, valves open. Warning signals can be generated in the following modes:

- Manual and automatic mode for motors, valves and dosers (Page 82) (input parameter `WarnTiMan`)
- Manual and automatic mode for motors, valves and dosers (Page 82) (input parameter `WarnTiAut`)

You specify the warning times in seconds using the input parameters `WarnTiMan` and `WarnTiAut` . If, for example, a valve opens, this is displayed at the output parameter with `WarnAct` = 1. The valve then opens after the set warning time has expired and `WarnAct` then returns to 0.

A corresponding warning is not outputted if the values specified for the warning times (`WarnTiMan` or `WarnTiAut`) are lower than the `SampleTime` parameter.

**Note**

The warning is activated for each actuation that causes the motor to start, even if this means that the valve is moved to the neutral position.

A parameterized warning time (`WarnTiMan`, `WarnTiAut`) is not active in the position control mode.

### Simulating signals

This block provides the standard function, Simulating signals (Page 64).

You can simulate the following values:

- Additional value (`SimAV`, `SimAV_Li`)

With internal simulation with immediate tracking of feedback, it is possible to simulate a position between the open and closed state (`FbkOpenOut` = `FbkCloseOut` = 0) by means of a stop command.

### Select unit of measure

This block provides the standard function, Selecting a unit of measure (Page 240).

### Neutral position

This block provides the standard function, Neutral position for motors, valves and controllers (Page 54).

### Output signal as pulse signal or static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 57). In addition to the static control outputs `Open` and `Close`, the block also has pulse outputs `P_Open`, `P_Close`, and `P_Stop`, which are dependent on the static control output.

### Generate instance-specific messages

This block provides the standard function, Generating instance-specific messages (Page 230).

### Resynchronization

This block provides the standard function Resynchronization (Page 258).

Resynchronization is switched on over feature bits:

| | |
|---|---|
| Feature2 Bit17 =1 | Resynchronization after CPU startup |
| Feature2 Bit18 =1 | Resynchronization after Simulation or after Out of service |
| Feature2 Bit19 =1 | Resynchronization after control system fault |
| Feature2 Bit20 =1 | Resynchronization with 0->1 trigger at StateTrkOn input |

If there are no restrictions the control outputs are tracked as follow:

**Valve position**

**NoFbkOpen =0, NoFbkClose =0:**

Valve position open: Status7.Bit2 := FbkOpen AND (Rbk >= PosDiOpen);

Valve position close: Status7.Bit3 := FbkClose AND (Rbk <= PosDiClose);

valve position intermediate: Status7.Bit4 := NOT Status7.Bit2 AND NOT Status7.Bit3;

If both valve feedback signals are not available (NoFbkOpen =1, NoFbkClose =1) only the readback value is relevant. The analog readback value is always available at VlvPosL.

**Position control is off (PosAct =0):**

**Motor controls**

**Motor feedbacks are present** Feature Bit 12 =0:

Open := FbkOpening AND NOT FbkClosing AND valve position is not open;

Close := FbkClosing AND NOT FbkOpening AND valve position is not close;

StopCtrl := NOT FbkOpening AND NOT FbkClosing;

In case if there are no plausible motor feedbacks (FbkOpening =1, FbkClosing =1) the motor goes to stop control.

**Motor feedbacks are not present** Feature Bit 12 =1:

The motor goes to stop control:

Open := false;

Close := false;

StopCtrl := true;

**Analog Readback value:**

MV_Out := Rbk; MV_Out is tracked and limited to MV_Scale.

**Position control is on (PosAct =1):**

Motor controls:

Depending on the resulting difference “MV_Out – Rbk” during “Tracking of State” the control outputs Open, Close, P_Open, P_Close are calculated.

Analog Readback value:

MV_Out := Rbk;

MV_Out is tracked and limited to MV_HiLim and MV_LoLim.

### Configurable reactions using the `Feature` parameter

An overview of all the reactions that are provided by the `Feature` parameter is available in section Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 3 | Enabling resetting of commands for the control settings (Page 172) |
| 4 | Setting switch or button mode (Page 177) |
| 8 | Sealing the valve (Page 187) |
| 9 | Resetting via input signals in the event of interlocking (Protection) or errors (Page 174) |
| 10 | Exiting local mode (Page 189) |
| 11 | Activating the run time of feedback signals (Page 164) |
| 12 | Motor feedback is not available (Page 168) |
| 13 | Separate monitoring time for stopping the motor (Page 180) |
| 14 | Enabling rapid stop via faceplate (Page 179) |
| 17 | Enabling bumpless switchover to automatic mode for valves, motors, and dosers (Page 184) |
| 18 | Activating error state for external process control error CSF (Page 162) |
| 19 | Reset even with locked state (Page 176) |
| 20 | Disable calculation of impulse control in LocalSetting 2 and 4 (Page 200) |
| 21 | Enable bumpless switchover to "Automatic" mode for operator only (Page 183) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 23 | Limits disabled in faceplate (Page 206) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 27 | Interlock display with LocalSetting 2 or 4 (Page 189) |
| 28 | Disabling operating points (Page 156) |
| 29 | Signaling limit violation (Page 181) |
| 30 | Set reset depending on the operating mode or the LiOp parameter (Page 174) |
| 31 | Activating reset of protection / error in manual mode (Page 175) |

In switching mode (`Feature.Bit4` = 1), the control is selected with the static signals `OpenAut` and `CloseAut`. If the `OpenAut` and `CloseAut` inputs are not set, the motor is stopped. Control via `StopAut` is not required. If the "Enable reset of commands for control" function

(`Feature.Bit3` = 1) is also enabled, the `OpenAut` and `CloseAut` inputs are reset to 0 after evaluation in the block.

In switching mode (`Feature.Bit4` = 1), the `PosOnAut` control acts as a switch. The "Enable resetting of commands for control" function (`Feature.Bit3` = 1) has no influence on `PosOnAut`.

### Display auxiliary values

This block provides the standard function, Displaying auxiliary values (Page 239).

### Configurable reactions using the `Feature2` parameter

| Bit | Function |
|---|---|
| 1 | Changes signal status of outputs in OOS (Page 208) |
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 3 | Control priority in the event of invalid or not plausible input commands (Page 186) |
| 4 | Setting switch or button mode for local commands (Page 195) |
| 5 | Evaluation of the signal status of the interlock signals (Page 153) |
| 6 | Vibrate after torque monitoring (Page 190) |
| 8 | Forcing commands in the Local mode (Page 198) |
| 9 | Button mode with triggered local control commands (Page 209) |
| 10 | Considering bad quality of automatic commands or external values (Page 200) |
| 11 | Suppress MsgLock and “Out of service” mode for a connected message block (Page 202) |
| 12 | Disable limit act output when limit value message is disabled (Page 209) |
| 13 | Resetting the Pending Error in Local Mode 1 and 3 (Page 210) |
| 16 | Separate interlock for each direction, position, or speed (Page 180) |
| 17 | Resynchronization after startup (Page 213) |
| 18 | Resynchronization after out of service or after simulation (Page 213) |
| 19 | Resynchronization after CSF (Page 214) |
| 20 | Resynchronization with StateTrkOn input (Page 214) |

### Connection of the time-stamped messages from EventTs or Event16Ts

This block provides the standard function Connection of the time-stamped messages from EventTs or Event16Ts (Page 233).

### SIMATIC BATCH functionality

This block provides the standard function, SIMATIC BATCH functionality (Page 73).

### Disable feedback

This block provides the standard function, Disabling feedback for valves (Page 106). Feedback monitoring can be disabled separately for each feedback signal with `NoFbkOpen` or `NoFbkClose`.

### Labeling of buttons

This block provides the standard function, Labeling of buttons and text (Page 235).

Instance-specific text can be configured for the following parameters:

- `OpenMan`
- `CloseMan`
- `StopMan`
- `RapidStp`

### Outputs for status of drive

The block has the following outputs to indicate the status of the drives. In the CFC plan, this information can be further interconnected:

| | |
|---|---|
| Closed =1 | Motor is stopped:<br>`Open`, `Close`, `FbkClsgOut`, `FbkOpngOut` =0<br>Valve is in position closed:<br>(`RbkOut` <= `PosDiClose`) AND `FbkClose` =1, `MV_Out` <= `PosDiClose` |
| Opened =1 | Motor is stopped:<br>`Open`, `Close`, `FbkClsgOut`, `FbkOpngOut` =0<br>Valve is in position opened:<br>(`RbkOut` >= `PosDiOpen`) AND `FbkOpen` =1, `MV_Out` >= `PosDiOpen` |
| Closing =1 | Motor drives in direction of position closed:<br>`Close`, `FbkClsgOut` =1 |
| Opening =1 | Motor drives in direction of position opened:<br>`Open`, `FbkOpngOut` =1 |
| Stop =1 | Motor is stopped and position is reached:<br>`Open`, `Close`, `FbkClsgOut`, `FbkOpngOut` =0 |

**Note**

The outputs for status of drive are only can be active if there is no monitoring error:

`MonDynErr` =0, `MonStaErr` =0

### See also

EventTs functions (Page 2046)

Description of VlvPosL (Page 1876)

VlvPosL messaging (Page 1910)

VlvPosL I/Os (Page 1912)

VlvPosL block diagram (Page 1928)

VlvPosL modes (Page 1882)

Tracking and limiting a manipulated variable (Page 222)

## 8.14.4 VlvPosL error handling

### Error handling of VlvPosL

For troubleshooting all blocks, see also the Error handling (Page 128) section in the basics.

The following errors can be displayed for this block:

- Error numbers
- Mode switchover error
- Invalid or not plausible input commands
- Control system fault (`CSF`)

### Overview of error numbers

The `ErrorNum` parameter can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Pre-defined value when inserting the block; the block is not executed. |
| 0 | No error pending. |
| 41 | The value for the `LocalSetting` connection is not within the valid limit of 0 to 4. |
| 42 | `LocalSetting` = 0 or `LocalSetting` = 3 or `LocalSetting` = 4 and `LocalLi` = 1 |
| 51 | `OpenLocal` = 1 and `StopLocal` = 1<br>`CloseLocal` = 1 and `StopLocal` = 1<br>`OpenLocal` = 1 and `CloseLocal` = 1<br>`OpenAut` = 1 and `StopAut` = 1<br>`CloseAut` = 1 and `StopAut` = 1<br>`OpenAut` = 1 and `CloseAut` = 1<br>`PosOnAut` = 1 and `StopAut` = 1<br>`PosOnAut` = 1 and `OpenAut` = 1<br>`PosOnAut` = 1 and `CloseAut` = 1<br>`AutModLi` = 1 and `ManModLi` = 1<br>`OpenForce` = 1 and `StopForce` = 1<br>`CloseForce` = 1 and `StopForce` = 1<br>`OpenForce` = 1 and `CloseForce` = 1 |
| 52 | `LocalAct` = 1 and `LocalSetting` = 2 or 4 and `SimOn` = 1 |

### Mode switchover error

This error can be outputted by the block, see section Error handling (Page 128).

### Invalid or not plausible input commands

This error can be outputted by the block, see section Error handling (Page 128).

For the following invalid or not plausible input commands, the control output can be stopped or switched to the neutral position. This depends on the function Control priority in the event of invalid or not plausible input commands (Page 186).

| Operating mode | Invalid or not plausible input commands | Control reaction with **Feature2.Bit3 = 1** |
|---|---|---|
| Local: Localsetting = 1<br>or Localsetting = 3 | Pushbutton operation for local mode (Feature2.Bit4 = 0):<br>OpenLocal = 1 and CloseLocal = 1 or<br>OpenLocal = 1 and StopLocal = 1 or<br>StopLocal = 1 and CloseLocal = 1<br>Switching mode (Feature2.Bit4 = 1):<br>OpenLocal = 1 and CloseLocal = 1 | Motor is started in valve neutral position direction. |
| Local: Localsetting = 1<br>or Localsetting = 3 and forcing | OpenForce = 1 and CloseForce = 1 or<br>OpenForce = 1 and StopForce = 1 or<br>StopForce = 1 and CloseForce = 1 | |
| Forcing and no "Local mode" | OpenForce = 1 and CloseForce = 1 or<br>OpenForce = 1 and StopForce = 1 or<br>StopForce = 1 and CloseForce = 1 | |
| "Automatic mode" and no forcing | Pushbutton operation (Feature.Bit4 = 0):<br>OpenAut = 1 and CloseAut = 1 or<br>OpenAut = 1 and StopAut = 1 or<br>StopAut = 1 and CloseAut = 1 or<br>StopAut = 1 and PosOnAut = 1 or<br>CloseAut = 1 and PosOnAut = 1 or<br>OpenAut = 1 and PosOnAut = 1<br><br>Switch mode (Feature.Bit4 = 1):<br>OpenAut = 1 and CloseAut = 1 or<br>CloseAut = 1 and PosOnAut = 1 or<br>OpenAut = 1 and PosOnAut = 1 | |
| "Manual mode" and no forcing | OpenMan = 1 and CloseMan = 1 or<br>OpenMan = 1 and StopMan = 1 or<br>StopMan = 1 and CloseMan = 1 | |

### Control system fault (CSF)

An external signal can be activated via the CSF input. If this signal CSF.Value = 1, a control system fault is triggered. For more information on this, see section Error handling (Page 128).

### See also

VlvPosL block diagram (Page 1928)

VlvPosL I/Os (Page 1912)

VlvPosL functions (Page 1884)

VlvPosL modes (Page 1882)

Description of VlvPosL (Page 1876)

## 8.14.5 VlvPosL messaging

### Message characteristics

The following messages can be generated for this block:

- Control system fault
- Process messages
- Instance-specific messages

### Control system fault

The following control system fault messages can be outputted:

| Message instance | Message ID | Message class | Event |
|---|---|---|---|
| `MsgEvId1` | SIG1 | AS control system message - fault | $$BlockComment$$<br>Motor feedback error |
| | SIG2 | AS control system message - fault | $$BlockComment$$<br>Motor protection triggered |
| | SIG3 | AS control system message - fault | $$BlockComment$$<br>Valve feedback error |
| | SIG4 | AS control system message - fault | $$BlockComment$$<br>External fault has occurred |

**Explanation**:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to the input parameter `CSF`. If this `CSF` = 1, a control system fault is triggered (`MsgEvId1`, SIG4).

### Process messages

| Message instance | Message ID | Message class | Event |
|---|---|---|---|
| MsgEvId2 | SIG1 | Alarm - high | $$BlockComment$$<br>ER - Alarm high limit violated |
| | SIG2 | Alarm - low | $$BlockComment$$<br>ER - Alarm low limit violated |
| | SIG3 | Warning - high | $$BlockComment$$<br>Rbk - High warning limit violated |
| | SIG4 | Warning - low | $$BlockComment$$<br>Rbk - Low warning limit violated |

### Instance-specific messages

You can use up to four instance-specific messages with this block.

| Message instance | Message ID | Message class | Event |
|---|---|---|---|
| MsgEvId1 | SIG5 | AS control system message - fault | $$BlockComment$$<br>External message 1 |
| | SIG6 | AS control system message - fault | $$BlockComment$$<br>External message 2 |
| MsgEvId2 | SIG5 | AS control system message - fault | $$BlockComment$$<br>External message 3 |
| | SIG6 | AS control system message - fault | $$BlockComment$$<br>External message 4 |

**Explanation**:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for the message instance **MsgEvId1**

| Associated value | Block parameter |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |
| 4 | ExtVa104 |
| 5 | ExtVa105 |
| 6 | ExtVa106 |
| 7 | Reserved |
| 8 | Reserved |
| 9 | Reserved |
| 10 | Reserved |

The associated values 4 to 6 are assigned to the parameters ExtVa104 to ExtVa106, and are usable. For more information on this, refer to the manual "*Process Control System PCS 7 - Engineering System*".

### Associated values for the message instance `MsgEvId2`

| Associated value | Block parameter |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | `ExtVa204` |
| 5 | `ExtVa205` |
| 6 | `ExtVa206` |
| 7 | Reserved |
| 8 | Reserved |
| 9 | Reserved |
| 10 | Reserved |

The associated values 4 to 6 are assigned to the parameters `ExtVa204` to `ExtVa206`, and are usable. For more information on this, refer to the manual "*Process Control System PCS 7 - Engineering System*".

### See also

Description of VlvPosL (Page 1876)

VlvPosL functions (Page 1884)

VlvPosL I/Os (Page 1912)

VlvPosL block diagram (Page 1928)

VlvPosL modes (Page 1882)

## 8.14.6 VlvPosL I/Os

I/Os of VlvPosL

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `AutModLi*` | 1= "Automatic mode" via interconnection or SFC (controlled via `ModLiOp` = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `AutModOp*` | 1 = "Automatic mode" via operator (controlled via `ModLiOp` = 1) | BOOL | 0 |
| `AV` | Input additional analog value, to be connected to `AV_Tech` of the `AV` block | ANY | |
| `AV_AH_Lim` | Limit high alarm | REAL | 95.0 |
| `AV_AL_Lim` | Limit low alarm | REAL | 5.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| AV_Hyst | Hysteresis for alarm, warning and tolerance limits | REAL | 1.0 |
| AV_TH_Lim | Limit high tolerance | REAL | 85.0 |
| AV_TL_Lim | Limit low tolerance | REAL | 15.0 |
| AV_WH_Lim | Limit high warning | REAL | 90.0 |
| AV_WL_Lim | Limit low warning | REAL | 10.0 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7 string | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BlkTypeData* | This input can not be edit or linked and is for internal usage only! | | |
| BreakTime* | Minimum break duration [s] | REAL | 1.0 |
| BypProt | 1 = Bypass interlock is enabled in "Local mode" and in simulation | BOOL | 0 |
| CloseAut* | 1 = Select Close valve in "Automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CloseChnST | Signal status of the output channel of Close.<br>It should be connected to an output channel block. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| CloseForce | 1 = Force Close valve | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CloseLocal | 1 = Select Close valve in "Local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CloseMan* | 1 = Select Close valve in "Manual mode" | BOOL | 0 |
| CSF | 1 = External fault (control system fault) Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DeadBand | Deadband for forming the output ER | REAL | 0.1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block is executed | BOOL | 1 |
| ER_A_DC* | Delay for incoming alarms during monitoring of the manipulated variable difference | REAL | 0.0 |
| ER_A_DG* | Delay for outgoing alarms during monitoring of the manipulated variable difference | REAL | 0.0 |
| ER_AH_En | 1 = Activate alarm (high) for manipulated variable difference monitoring | BOOL | 1 |
| ER_AH_Lim | Alarm limit (high) for manipulated variable difference monitoring | REAL | 100.0 |
| ER_AH_MsgEn | 1 = Enable messages for alarm (high) for manipulated variable difference monitoring | BOOL | 1 |
| ER_AL_Lim | Alarm limit (low) for manipulated variable difference monitoring | REAL | -100.0 |
| ER_AL_En | 1 = Activate alarm (low) for manipulated variable difference monitoring | BOOL | 1 |
| ER_AL_MsgEn | 1 = Enable messages for alarm (low) for manipulated variable difference monitoring | BOOL | 1 |
| ER_Hyst | Alarm hysteresis for manipulated variable difference monitoring | REAL | 1.0 |
| EventTsIn | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>Input parameter EventTsIn is used to interconnect output parameter EventTsOut of the EventTs, Event16TS block. When this interconnection is configured, the messages of the EventTs, Event16TS block are displayed and can also be acknowledged on the OS in the message view of the technological block. | ANY | |
| EventTs2In | For interconnecting data between a technology block and the message blocks EventTs, Event16Ts.<br>The EventTs2In input parameter serves to interconnect the EventTsOut output parameter of the EventTs, Event16Ts block. When this interconnection is configured, the messages of the EventTs, Event16Ts block are displayed on the OS in the alarm view of the technology block and can also be acknowledged there. | ANY | |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg3 | Binary input for freely selectable message 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg4 | Binary input for freely selectable message 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa104 | Associated value 4 for messages (MsgEvId1) | ANY | |
| ExtVa105 | Associated value 5 for messages (MsgEvId1) | ANY | |
| ExtVa106 | Associated value 6 for messages (MsgEvId1) | ANY | |
| ExtVa204 | Associated value 4 for messages (MsgEvId2) | ANY | |
| ExtVa205 | Associated value 5 for messages (MsgEvId2) | ANY | |
| ExtVa206 | Associated value 6 for messages (MsgEvId2) | ANY | |
| FaultExt | 1 = External fault Error handling (Page 128) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkClose | 1 = Valve closed feedback signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| FbkClosing | 1 = Valve closing feedback signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| FbkOpen | 1 = Valve open feedback signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| FbkOpening | 1 = Valve opening feedback signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| Feature | I/O for additional functions (Page 1884) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Feature2 | I/O for additional functions (Page 1884) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| Gain | Position control gain | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#FF |
| Intlock | 0 = Interlock / Interlock Open is activated.<br>Once the interlock condition has cleared, the block can be operated without reset.<br>1 = Interlock / Interlock Open is deactivated. | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| Intl_En | 1 = Parameter Intlock is enabled | BOOL | 1 |
| IntlCls | 0 = Interlock Close is activated.<br>Once the interlock condition has cleared, the block can be operated without reset.<br>1 = Interlock Close is deactivated. | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| IntlClsEn | 1 = Parameter IntlCls is enabled | BOOL | 1 |
| LocalLi | 1 = Activate "Local mode" via plant signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalOp* | 1 = "Local mode" via operator | BOOL | 0 |
| LocalSetting | Properties for Local mode (Page 86) | INT | 0 |
| Man* | Manipulated variable in "manual mode" | REAL | 0.0 |
| ManModLi* | 1 = "Manual mode" via interconnection or SFC (controlled via ModLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManModOp* | 1= "Manual mode" via: OS operator (controlled via ModLiOp = 0) | BOOL | 1 |
| ModLiOp | Toggle operation between:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MonSafePos | 1 = Go to neutral position in the event of monitoring errors | BOOL | 1 |
| MotorTime* | Motor actuating time [s] | REAL | 30.0 |
| M_Monitor | 1 = Motor feedback monitoring | BOOL | 1 |
| M_MonTiDynamic* | Monitoring time for feedback errors or feedback start errors after successful operation in [s] | REAL | 3.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| M_MonTiStatic* | Monitoring time for feedback errors without operation in [s] | REAL | 3.0 |
| M_MonTiStop* | Monitoring time for feedback stop errors after successful operation in [s] | REAL | 3.0 |
| MS_RelOp* | 1= Release for maintenance via OS operator | BOOL | 0 |
| MsgEvId1 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgEvId2 | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>For more on this, refer to the section Suppressing messages using the MsgLock parameter (Page 231). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV | Manipulated variable in "automatic mode" | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_Forced | Forced manipulated variable that is not limited and has top priority | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_ForOn | 1 = Output forced manipulated variable MV_Forced unlimited at output MV | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_HiLim | Limit (high) for manipulated variable MV | REAL | 100.0 |
| MV_LoLim | Limit (low) for manipulated variable MV | REAL | 0.0 |
| MV_OpScale | OS display range for the manipulated variable MV | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| MV_Scale | Range for the manipulated variable MV | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| MV_Trk | Tracking value for manipulated variable MV | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_TrkOn | 1 = Tracking of manipulated variable MV | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_Unit | Unit of measure for manipulated variable | INT | 1342 |
| NoFbkClose | 1 = No feedback present for "valve closed" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NoFbkOpen | 1 = No feedback present for "valve open" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Occupied | 1 = Occupied by a batch | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpenAut* | 1 = Select Open valve in "Automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpenChnST | Signal status of the output channel of Open.<br>It should be connected to an output channel block. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| OpenForce | 1 = Force Open valve | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpenLocal | 1 = Select Open valve in "Local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpenMan* | 1 = Select Open valve in "Manual mode" | BOOL | 0 |
| OpSt_In | Input parameter for local operator permission, to be connected with the Out output parameter of the upstream block OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for Operator control permissions (Page 285) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| OS1Perm | I/O for Operator control permissions (Page 285) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| Permit | 1 = Activation / activation Open is activated<br>0 = Activation / activation Open is deactivated | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| PermitSafePos | 1=Valve SafePos / Stop allowed | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PermSf_En | 1=Permit SafePos / Stop enabled | BOOL | 1 |
| Perm_En | 1 = Parameter Permit is enabled | BOOL | 1 |
| PermCls | 1 = Activation Close is activated<br>0 = Activation Close is deactivated | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| PermClsEn | 1 = Parameter PermCls is enabled | BOOL | 1 |
| PosDiClose | Limit for control valve position "closed" | REAL | 5.0 |
| PosDiOpen | Limit for control valve position "open" | REAL | 95.0 |
| PosOnAut | 1 = Select Activate position control in "Automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PosOnMan | 1 = Select Activate position control in "Manual mode" | BOOL | 0 |
| PosRbkHys | allowable variation from preset position;<br>should be greater than the parameter DeadBand | REAL | 0.1 |
| Protect | 0 = Protective interlock / protective interlock Open is activated.<br>Once the interlock condition has cleared, the block must be reset.<br>1 = Protective interlock / protective interlock Open is deactivated. | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| Prot_En | 1 = Parameter Protect is enabled | BOOL | 1 |
| ProtCls | 0 = Protective interlock Close is activated.<br>Once the interlock condition has cleared, the block must be reset.<br>1 = Protective interlock Close is deactivated. | STRUCT<br>• Value:BOOL<br>• ST:BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 1<br>• 16#FF<br>• 0 |
| ProtClsEn | 1 = Parameter ProtCls is enabled | BOOL | 1 |
| PulseTime | Minimum pulse duration [s] | REAL | 1.0 |
| PulseWidth* | Pulse width of the control signal [s] | REAL | 3.0 |
| RapidStp* | Rapid stop for the motor<br>0 = Motor On<br>1 = Motor off | BOOL | 0 |
| Rbk | Position feedback | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RbkHyst | Alarm hysteresis for position feedback | REAL | 1.0 |
| RbkMonitor | 1 = Position feedback monitoring | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RbkMonTi | Monitoring time for the position feedback in [s] | REAL | 5.0 |
| RbkW_DC* | Delay time for incoming warnings [s] | REAL | 0.0 |
| RbkW_DG* | Delay time for outgoing warnings [s] | REAL | 0.0 |
| RbkWH_En | 1 = Enable warning (high) for position feedback | BOOL | 0 |
| RbkWH_Lim | Limit for position feedback of warning (high) | REAL | 90.0 |
| RbkWH_MsgEn | 1 = Enable messages for warning (high) for position feedback | BOOL | 1 |
| RbkWL_En | 1 = Enable warning (low) for position feedback | BOOL | 0 |
| RbkWL_Lim | Limit for position feedback of warning (low) | REAL | 10.0 |
| RbkWL_MsgEn | 1 = Enable messages for warning (low) for position feedback | BOOL | 1 |
| RstLi* | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | 1 = Reset via operator | BOOL | 0 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SafePos | Neutral position for valve:<br>0 = Closed<br>1 = Open<br>2 = Stop (valve can be in position Closed, Opened or Intermediate) | INT | 2 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | Call a block saved in this parameter as an Opening additional faceplates (Page 234) in the standard view | ANY | - |
| SelFp2 | Call a block saved in this parameter as an Opening additional faceplates (Page 234) in the preview | ANY | - |
| SimAV* | Additional value used with SimOn = 1 | REAL | 0.0 |
| SimAV_Li | Additional analog value used with SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled via SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimRbk* | Position feedback used with SimOn = 1 | REAL | 0.0 |
| SimRbkLi | Position feedback used with SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| StopAut* | 1 = Stop the motor in "Automatic mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopForce | 1 = Force motor stop | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopLocal | 1 = Stop the motor in "Local mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StopMan* | 1 = Stop the motor in "Manual mode" | BOOL | 0 |
| TmLag | Delay time of the LeadLag filter [s] | REAL | 0.0 |
| TorqClose | 0 = Torque shutdown active when closing<br>1 = Good state | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| TorqOpen | 0 = Torque shutdown active when opening<br>1 = Good state | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| Trip | 1 = Motor is in "good" state | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#FF |
| UA1unit | Unit of measure for analog auxiliary value 1 | INT | 0 |
| UA2unit | Unit of measure for analog auxiliary value 2 | INT | 0 |
| UserAna1 | Analog auxiliary value 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UserAna2 | Analog auxiliary value 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |
| VibrNo | Vibration number if TorqOpen or TorqClose | INT | 3 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| VibrPerc | Vibration Width [%] if TorqOpen or TorqClose (in analog mode) | REAL | 5.0 |
| VibrWidth | Vibration Width [s] if TorqOpen or TorqClose (in binary mode) | REAL | 3.0 |
| V_Monitor | 1 = Valve feedback monitoring | BOOL | 1 |
| V_MonTiDynamic* | Valve monitoring time after operation in [s] | REAL | 30.0 |
| V_MonTiStatic* | Monitoring time for valve feedback errors without operation in [s] | REAL | 5.0 |
| WarnTiAut* | Prewarning of valve movement in "automatic mode" in [s] | REAL | 0.0 |
| StateTrkOn | Initiate tracking of state | BOOL | 0 |
| WarnTiMan* | Prewarning of valve movement in "manual mode" in [s] | REAL | 0.0 |

* Values can be written back to these inputs during execution of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AutAct | 1 = "Automatic mode" enabled<br>0 = "Manual mode" is enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AutoST | 1 = Worst signal status of automatic commands | BYTE | 16#80 |
| AV_OpScale | Limit for scale in AV bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| AV_Out | Output additional analog value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| AV_Unit | Unit of measure for additional analog value | INT | 0 |
| CascaCut | Cascade connection:<br>1 = Control chain from master controller to secondary valve is interrupted | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Close | Control output<br>1= Close valve | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Closed | 1 = Valve is closed | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Closing | 1 = Valve is closing | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| CurrMonM | Remaining motor monitoring time [s] | DINT | 0 |
| CurrMonRbk | Remaining monitoring time of position feedback [s] | DINT | 0 |
| CurrMonV | Remaining valve monitoring time [s] | DINT | 0 |
| CSF_Out | CSF as Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ER | Difference of the manipulated variable MV_Out and the position feedback RbkOut in consideration of the parameter DeadBand | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ER_AH_Act | 1 = Alarm limit (high) for manipulated variable difference violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ER_AL_Act | 1 = Alarm limit (low) for manipulated variable difference violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ER_AH_LimOut | ER_AH_Lim as output | REAL | 0.0 |
| ER_AL_LimOut | ER_AL_Lim as output | REAL | 0.0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see the section VlvPosL error handling (Page 1908) | INT | -1 |
| FaultExtOut | FaultExt as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkCloseOut | Valve closed feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkClsgOut | Valve closing feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkOpenOut | Valve open feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FbkOpngOut | Valve opening feedback | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| GrpErr | 1 = Group error pending | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| IntlockOut | Intlock as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IntlClsOut | IntlCls as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LocalAct | 1 = "Local mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LockAct | 1 = Interlock (Intlock, Permit, Protect) or Trip is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManAct | 1 = "Manual mode" enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| M_MonDynErr | 1 = Feedback error or feedback start error of the motor due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| M_MonStaErr | 1 = Motor feedback error due to unexpected feedback change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| M_MonStopErr | 1 = Feedback stop error of the motor due to unexpected feedback change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Release | Release for maintenance:<br>1 = Enable for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1 | Message - acknowledgment status 1 (ACK_STATE output of the first ALARM_8P) | WORD | 16#0000 |
| MsgAckn2 | Message - acknowledgment status 2 (ACK_STATE output of the first ALARM_8P) | WORD | 16#0000 |
| MsgErr1 | Message error 1 (ERROR output of the first ALARM_8P) | BOOL | 0 |
| MsgErr2 | Message error 2 (ERROR output of the first ALARM_8P) | BOOL | 0 |
| MsgStat1 | Message status 1 (STATUS output of the first ALARM_8P) | WORD | 16#0000 |
| MsgStat2 | Message status 2 (STATUS output of the first ALARM_8P) | WORD | 16#0000 |
| MV_HiAct | 1 = Limit (high) of manipulated variable violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MV_LoAct | 1 = Limit (low) of manipulated variable violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MV_Aut | Manipulated variable in automatic mode | REAL | 0.0 |
| MV_Out | Manipulated variable | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| MV_UnitOut | Unit of measure for manipulated variable | INT | 0 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Open | Control output:<br>1 = Open the valve | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Opened | 1 = Valve is open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Opening | 1 = Valve is opening | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the input parameter OpSt_In for further interconnection with other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS1PermLog | Display of OS1Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| PosAct | Position control is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PosReached | 1 = Control valve has reached specified position | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Close | 1 = Pulse signal to close valve | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Open | 1 = Pulse signal to open valve | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| P_Rst | 1= Pulse output for reset<br>The parameter persists for one cycle after a reset. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Stop | 0 = Pulse signal for stopping the valve | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| PermitOut | Permit as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PermitSafePos-Out | PermitSafePos as output | STRUCT<br>• Value: BOOL<br>ST: BYTE | -<br>• 1<br>• 16#FF |
| PermClsOut | PermCls as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ProtectOut | Protect as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ProtClsOut | ProtCls as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| R_StpAct | 1 = Rapid stop of the motor is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RbkMonDynErr | 1 = Position feedback error due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RbkMonStaErr | 1 = Position feedback error due to unexpected feedback change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RbkOut | Position feedback output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RbkWH_Act | 1 = Warning (high) enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RbkWL_Act | 1 = Warning (low) enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RdyToReset | 1 = Ready for reset via RstLi input or commands in "local", "automatic", or "manual mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyToStart | 1 = Ready to start | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 1876) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1876) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1876) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 1876) | DWORD | 16#00000000 |
| Status5 | Status word 5 (Page 1876) | DWORD | 16#00000000 |
| Status6 | Status word 6 (Page 1876) | DWORD | 16#00000000 |
| Status7 | Status word 7 (Page 1876) | DWORD | 16#00000000 |
| Stop | 1 = Motor is stopped and position is reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SumMsgAct | 1 = Active hardware interrupt | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ThreshOff | Trip threshold of the three-step element | REAL | 0.0 |
| ThreshOn | Response threshold of the three-step element | REAL | 0.0 |
| TripOut | Trip as Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| UA1unitOut | UA1 as output | INT | 0 |
| UA2unitOut | UA2 as output | INT | 0 |
| UserAna1Out | UserAna1 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| UserAna2Out | UserAna2 as output | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| V_MonDynErr | 1 = Valve feedback error due to control change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| V_MonStaErr | 1 = Valve feedback error due to unexpected feedback change | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ResyncST | Worst Signal Status of inputs for resynchronization | BYTE | 16#80 |
| StopCtrl | Control output go to SafePos | BOOL | 0 |
| WarnAct | 1 = Prewarning for control valve movement away from active (WarnTiAut and WarnTiMan parameters) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### See also

VlvPosL messaging (Page 1910)
VlvPosL block diagram (Page 1928)
VlvPosL modes (Page 1882)
Disabling operating points (Page 156)
Signaling limit violation (Page 181)
Error handling (Page 128)

## 8.14.7 VlvPosL block diagram

### Block diagram of VlvPosL

A block diagram is not provided for this block.

### See also

VlvPosL I/Os (Page 1912)
VlvPosL messaging (Page 1910)
VlvPosL error handling (Page 1908)
VlvPosL functions (Page 1884)
VlvPosL modes (Page 1882)
Description of VlvPosL (Page 1876)

## 8.14.8 Operator control and monitoring

### 8.14.8.1 VlvPosL views

#### Views of the VlvPosL block

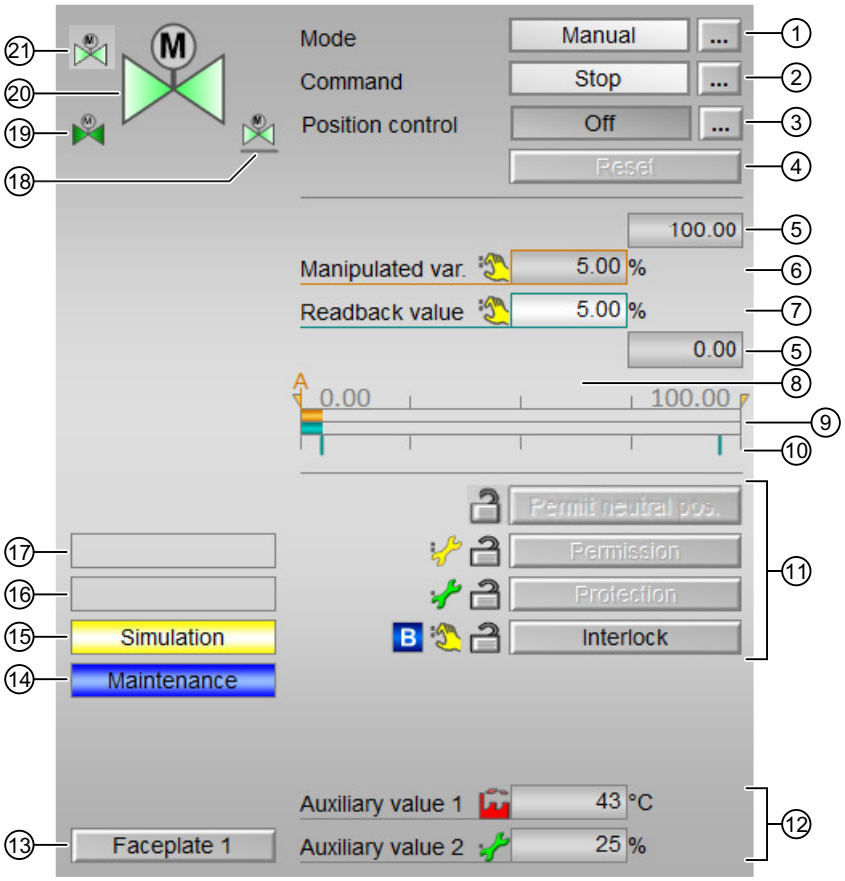
The VlvPosL block provides the following views:

- VlvPosL standard view (Page 1930)
- Limit view of motors (Page 328)
- VlvPosL limit value view (Page 1935)
- Alarm view (Page 336)
- Trend view (Page 341)
- VlvPosL parameter view (Page 1937)
- VlvPosL preview (Page 1941)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for VlvPosL (Page 1947)

You can find general information about the faceplate and block icon in the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260).

### 8.14.8.2 VlvPosL standard view

#### Standard view of VlvPosL




#### (1) Display and switch the operating mode

This area displays the current operating mode of the block. The following operating modes can be shown here:

- Manual and automatic mode for motors, valves and dosers (Page 82)
- Local mode (Page 86)
- Out of service (Page 77)

For information on switching the mode, see section Switching operating states and operating modes (Page 289).

### (2) Open, close and stop the motor valve

This area displays the default operating state for the motor valve. The following states can be shown and executed here:

- "Open"
- "Close"
- "Stop"
- "Rapid stop"

Analog positioning mode is terminated if one of these commands is executed. Positioning mode can be reactivated using the **(3)** button.

For information on switching the mode, see section Switching operating states and operating modes (Page 289).

If the text is configured for these commands, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (3) Activate positioning at an analog manipulated variable

This area shows you the default operating state for positioning to the analog manipulated variable. The following states can be shown here:

- "On"
- "Off"

Only the positioning mode is activated here:

- "On"

Positioning mode is terminated when the motor valve is operated using the **(2)** button.

### (4) Reset the block

Click "Reset" in the event of interlocks or errors. You can find more information on this in the section Resetting the block in case of interlocks or errors (Page 47).

### (5) High and low scale range for the manipulated variable

These values provide information on the display range for the bar graph of the manipulated variable. The scale range is defined in the engineering system.

### (6) Display and change the manipulated variable including signal status

This area shows the current manipulated variable with the corresponding signal status.

The manipulated variable can only be changed in manual mode and active position control.

You can find more information about changing the manipulated variable in the section Changing values (Page 291).

### (7) Display for position feedback including signal status

This area shows the current feedback of the manipulated variable with the corresponding signal status.

### (8) Display of the manipulated variable for automatic mode

The display [A] is only visible when the block is not in automatic mode. In Automatic mode, it is displayed when it needs to be reset or when tracking or forced tracking is enabled.

This display shows the manipulated variable that would apply if you were to switch to automatic mode or if you were to reset in automatic mode or disable tracking or forced tracking.

### (9) Bar display for manipulated variable

This area shows the current manipulated variable in the form of a bar graph. You can also see the high and low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (10) Bar display for position feedback

This area shows the current position feedback in the form of a bar graph. You can also see the high and low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

The limits for the "open" and "closed" positions are shown with 2 green lines.

For more information, refer: Bar graph (Page 376)

### (11) Area for operating the interlock functions of the block

This display is only visible when the corresponding block input is interconnected.

You can use these buttons to control the interlock functions of the block. You can find more information on this in the section Interlocking functions (Page 107).

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 113)), for example:

- Signal status (see Forming and outputting the signal status for technologic blocks (Page 117)), for example:

- Bypass information (see Forming the group status for interlock information (Page 113)):

If the text is configured for the interlocks, it is displayed as status text and as button labels for command selection. For more information, see section Labeling of buttons and text (Page 235).

### (12) Display for auxiliary values

This display is only visible when the corresponding block input is interconnected.

You can use this area to display two auxiliary values that have been configured in the engineering system. You can find more information on this in the section Displaying auxiliary values (Page 239).

### (13) Jump key to standard view of any faceplate

This display is only visible when the corresponding block input is interconnected.

Use this navigation button to jump to the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find more information on this in the section Opening additional faceplates (Page 234).

### (14) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

You can find more information on this in the section Release for maintenance (Page 70) display area for block states.

### (15) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"
- "Time delay"

You can find more information on this in the sections Simulating signals (Page 64) and Display of delay times (Page 288).

### (16) Display area for block states

This area provides additional information on the operating state of the block:

- "Motor protection"
- "External error"
- "Vibrate enabled"
- "End position error"
- "Control error"
- "Internal error"
- "Changeover error"
- "Torque active"
- "Invalid Signal"

You can find more information on this in the sections Monitoring the feedbacks (Page 104), Error handling (Page 128) (headings "Invalid input signals" and "Mode switchover error") and Motor protection function (Page 106).

### (17) Display area for block states

This area provides additional information on the operating state of the block:

- "Forced open" (`OpenForce`)
- "Forced closed" (`CloseForce`)
- "Forced stop" (`StopForce`)
- "Forced tracking" (`MV_ForOn`)
- "Tracking" (`MV_TrkOn`)
- "Request 0/1": Reset is expected but reset is not possible from faceplate. For more details, refer:Reset of errors during Request 0/1 (Page 382)

You can find more information on this in the section Forcing operating modes (Page 45).

### (18) Neutral position of the valve

This representation shows the neutral position for the valve:

- **Green**: Neutral position is "Open" or neutral position is “Stop” and valve is opened
- **Gray**: Neutral position is "Closed" or neutral position is “Stop” and valve is closed
- **Light green**: Neutral position is "Stop" and valve is in intermediate position

### (19) Automatic preview

This display is only visible in "Manual mode", in "Local mode", or with a reset request in "Automatic mode", when the current output signals are not identical to the control in "Automatic mode".

The display shows what state the valve would assume if you switched from "Manual" or "Local" mode to "Automatic mode", or performed a reset to "Automatic mode".

| Mode | Discription |
|---|---|
|  | Open: Automatic commands will open the motor valve. If position control is On or is switched on by automatic command, the Automatic `MV_Aut` is in open position. |
|  | Close: Automatic commands will close the motor valve. If position control is On or is switched on by automatic command, the Automatic `MV_Aut` is in close position. |
|  | Stop: Automatic commands will stop the motor valve or move it in another intermediate position. If position control is On or is switched on by automatic command, the Automatic `MV_Aut` is in intermediate position. |

The symbols are only visible if the position is not reached.

### (20) Status display of the motor valve

The current status of the motor valve is graphically displayed here.

You can find more information on this in the section Block icon for VlvPosL (Page 1947).

### (21) Mini preview for Resynchronization

Displays the mini preview for Resynchronizing the signal state.

The mini preview displays the following states:

- Open
- Close
- Intermediate

**Note**

Mini preview for resynch is visible only if resynchronization command is different to control output value.

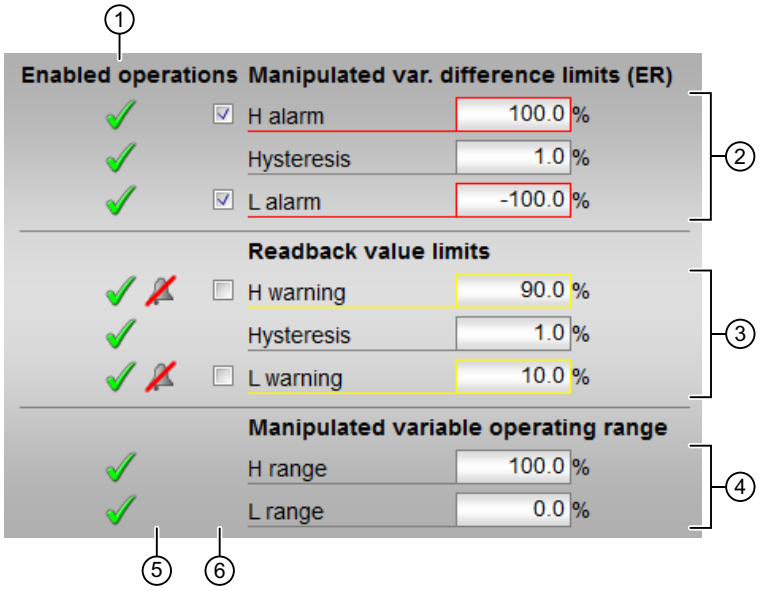
## 8.14.8.3 VlvPosL limit value view

### Limit view of VlvPosL

Several values are set in this view by default:

- Manipulated variable difference limits
- Readback value limits
- Manipulated variable operating range

The toolbar of the faceplate and the block icon indicate when the limits are reached or violated.




### (1) Enabled operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for operation enable:

- **Green check mark**: The OS operator can control this parameter
- **Gray check mark**: The OS operator cannot control this parameter at this time due to the process
- **Red cross**: The OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

### (2) Limits of the manipulated variable difference

In this area, you can enter the limits for the manipulated variable difference. For more on this, see section Changing values (Page 291).

You can change the following limits:

- "H alarm": Alarm high
- "Hysteresis"
- "L alarm": Alarm low

### (3) Readback value limits (`Rbk`)

In this area, you can enter the limits for the readback value (position feedback). For more on this, see section Changing values (Page 291).

You can change the following limits:

- "H warning": Warning high
- "Hysteresis"
- "L warning": Warning low

### (4) Manipulated variable operating range

In this area, you can enter the limits for the manipulated variable operation range. For more on this, see section Changing values (Page 291).

You can change the following limits:

- "H range": Range limit high
- "L range": Range limit low

### (5) Message suppression / delay

Message suppression indicates whether or not the suppression of the associated message in the AS block is activated with the `xx_MsgEn` parameter. The output of messages is not suppressed when you install the block (all `xx_MsgEn` parameters are assigned the default value 1). Messages can only be outputted if limit monitoring of the additional analog value has been enabled.

Alarm delays are also displayed in this position; for more on this, see section Area of application of the alarm delays (Page 225).

### (6) Suppress messages

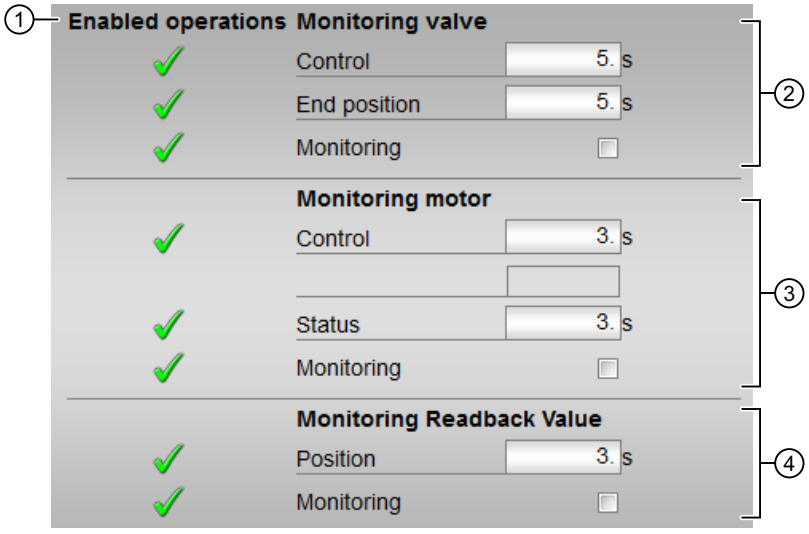
You can enable/disable messages by setting the check mark.

## 8.14.8.4 VlvPosL parameter view

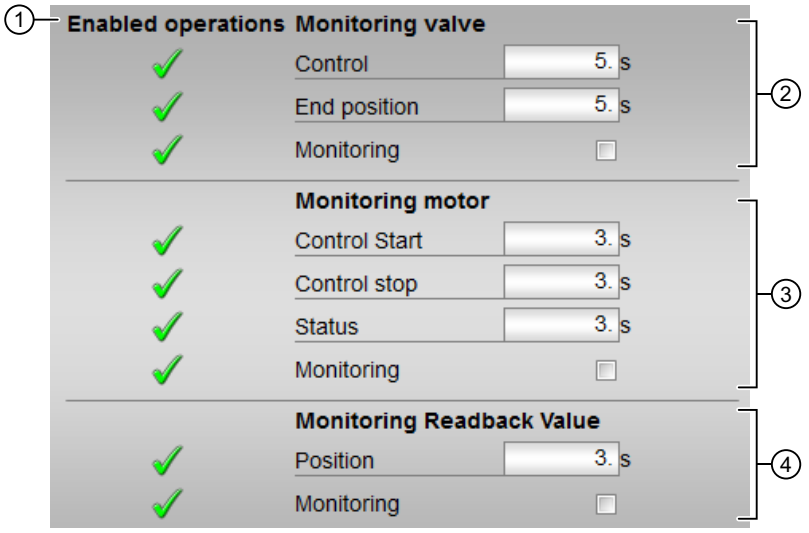
### Parameter view of VlvPosL

The parameter has an upper half and a lower half. You can navigate between the two halves with the arrow keys.

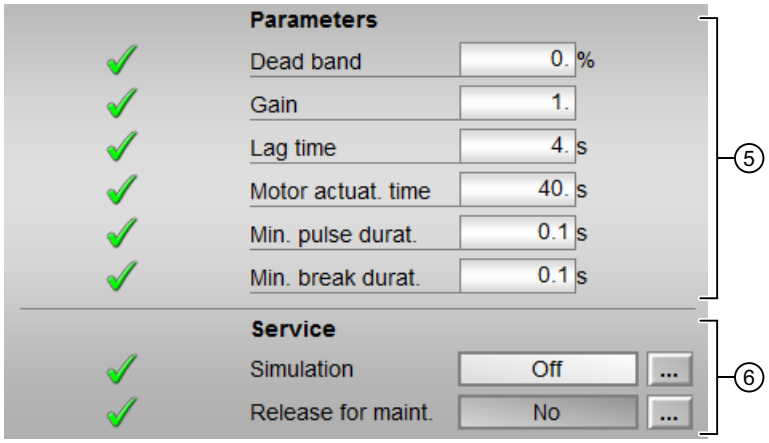
Upper half of the parameter view of VlvPosL with `Feature.Bit13` = 0:




Upper half of the parameter view of VlvPosL with `Feature.Bit13` = 1:




Lower half of the parameter view of VlvPosL:




## (1) Enabled operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for operation enable:

- **Green check mark**: The OS operator can control this parameter.
- **Gray check mark**: The OS operator cannot control this parameter at this time due to the process.
- **Red cross**: The OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

## (2) Monitor valve

In this area, you can change parameters and thereby influence the valve. You can find more information on this in the section Changing values (Page 291).

You can influence the following parameters:

- "Control": Monitoring time while "Opening"/"Closing" the valve (dynamic)
- "End position": Monitoring time for maintaining the valve position (static)

### Enable monitoring

You can enable monitoring by selecting the check box (☑).

You can find more information on this in the section Monitoring the feedbacks (Page 104).

## (3) Monitor motor

In this area, you change parameters and thereby influence the motor. You can find more information on this in the section Changing values (Page 291).

You can influence the following parameters:

- "Control": Monitoring time during startup and stopping of the motor (dynamic) `Feature.Bit13` = 0
- "Control start": Monitoring time during startup of the motor (dynamic) `Feature.Bit13` = 1
- "Control stop": Monitoring time during stopping of the motor (dynamic) `Feature.Bit13` = 1
- "Status": Monitoring time during permanent operation of the motor (static)

**Enable monitoring**

You can enable monitoring by selecting the check box (☑).

You can find more information on this in the section Monitoring the feedbacks (Page 104).

### (4) Monitoring the valve position

In this area, you change parameters and thereby influence the valve. You can find more information on this in the section Changing values (Page 291).

You can influence the following parameter:

- "Position": Monitoring time for the valve

**Enable monitoring**

You can enable monitoring by selecting the check box (☑).

You can find more information on this in the section Monitoring the feedbacks (Page 104).

### (5) "Parameters"

In this area, you change parameters and thereby influence the controller. For more on this, see section Changing values (Page 291).

You can influence the following parameters:

- "Deadband": Width of deadband
- "Gain": Gain
- "Delay": Delay time in [s]
- "Motor actuating time": Motor actuating time [s]
- "Minimum pulse duration": Minimum pulse duration [s]
- "Minimum break duration": Minimum break duration [s]

### (6) Service

You activate the following functions in this area:

- "Simulation"
- "Release for maintenance" (with display for maintenance demanded)

You can find more information on this in the section Switching operating states and operating modes (Page 289).

You can find information on this area in the sections:

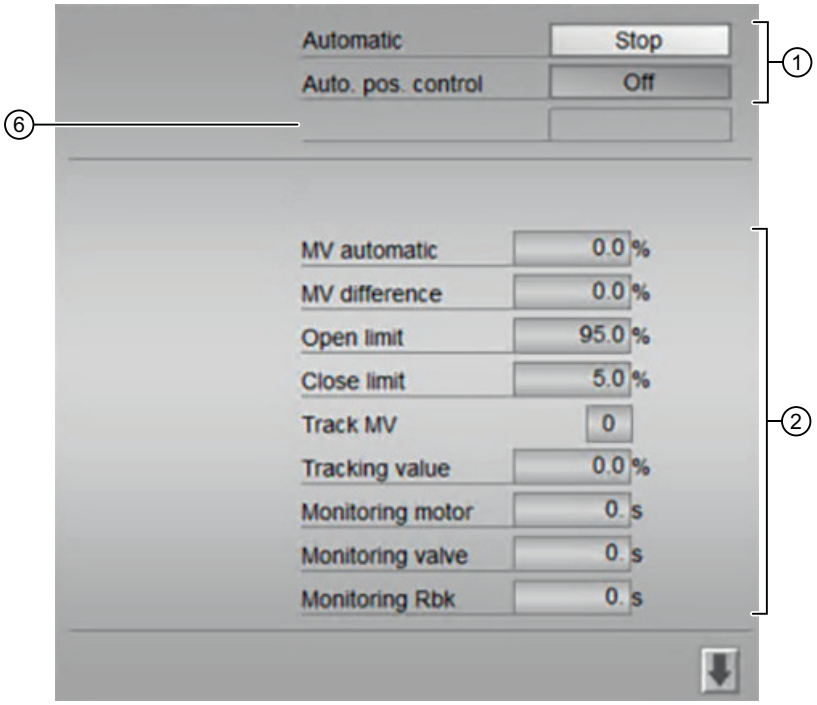
- Simulating signals (Page 64)
- Release for maintenance (Page 70)

## 8.14.8.5 VlvPosL preview

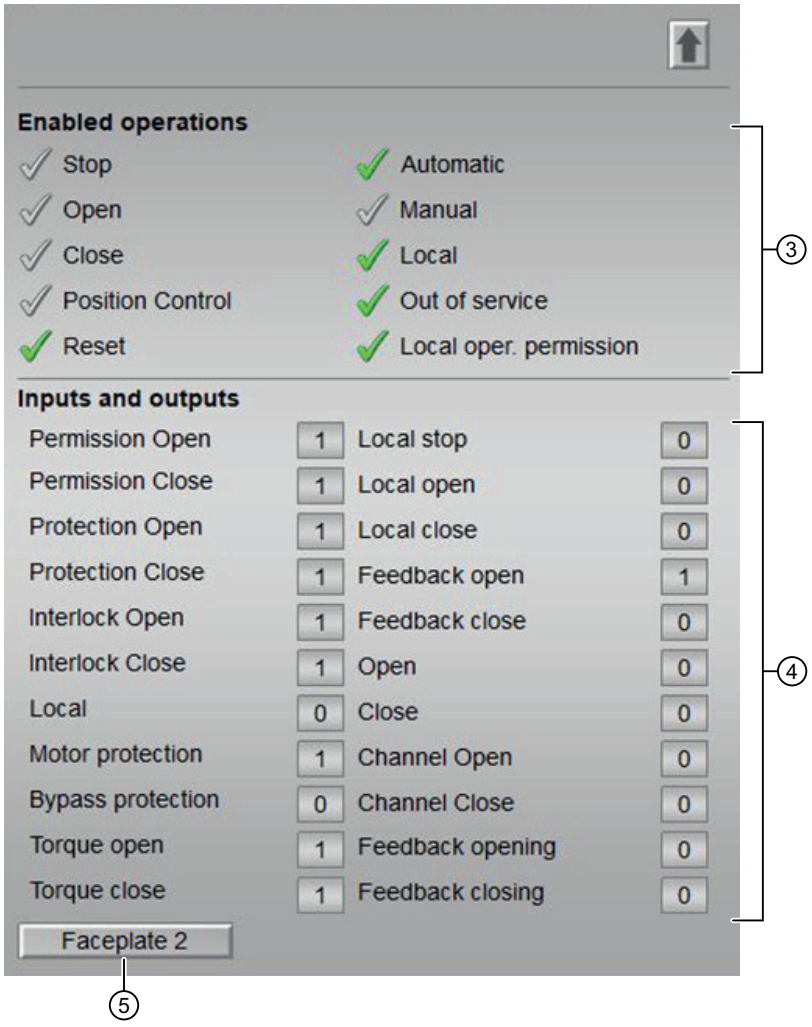
### Preview of VlvPosL

The preview has an upper half and a lower half. You can navigate between the two halves with the arrow keys.

Upper half of the preview of VlvPosL:




Lower half of the preview of VlvPosL:




## (1) Automatic preview

This area shows you the block status after it has switched from the "Manual" mode or "Local" mode to the "Automatic" mode.

| Mode | Discription |
|---|---|
| Open | Automatic commands will open the motor valve or valve is already opened. If position control is On or is switched on by automatic command, the automatic `MV_Aut` is in open position. |
| Close | Automatic commands will close the motor valve or valve is already closed. If position control is On or is switched on by automatic command, the Automatic `MV_Aut` is in close position. |
| Stop | Automatic commands will stop the motor valve or the automatic `MV_Aut` move it in another intermediate position. If position control is On or is switched on by automatic command, the automatic `MV_Aut` is in intermediate position. |

If the block is in "Automatic" mode, the current block state is displayed.

**Note**

- If the valve is closed and interlock locking is active (`SafePos`=0 Close), the automatic preview does not show the current status "Closed", instead it shows the active automatic command "Open".

This area also displays the worst signal status of the following automatic commands:

- `OpenAut`
- `CloseAut`
- `StopAut`
- `PosOnAut`
- `MV`

### (2) Preview area

- "MV Automatic": Display of the current automatic manipulated variable (`MV_Aut`). It depends on tracking (`MV_TrkOn, MV_Trk`) and is limited (`MV_HiLim, MV_LoLim`).
- "MV difference": Current manipulated variable difference (`ER`).
- "Open limit": Limit (`PosDiOpen`) for forming the "Control valve open" signal (`FbkOpenOut`). If the position feedback reaches this limit, the control valve is open.
- "Close limit": Limit (`PosDiClose`) for forming the "Control valve closed" signal (`FbkCloseOut`). If the position feedback reaches this limit, the control valve is closed.
- "Manipulated variable tracking": (`MV_TrkOn` = 1) Manipulated variable is tracked to the tracking value.
- "Tracking value": Effective manipulated variable for "Track manipulated variable"
- "Monitoring motor": Display of the remaining monitoring time of the motor. This display is only visible when monitoring is activated.
- "Monitor valve": Display of the remaining monitoring time of the valve. This display is only visible when monitoring is activated.
- "Monitoring Rbk": Display of the remaining monitoring time for the position feedback. This display is only visible when monitoring is activated.
- "Rbk Resync": Shows the input Rbk value for Resynchronization

### (3) Enabled operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for operation enable:

- **Green check mark**: The OS operator can control this parameter.
- **Gray check mark**: The OS operator cannot control this parameter at this time due to the process.
- **Red cross**: The OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

The following enabled operations are shown here:

- "Stop": You can stop the motor of the valve.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Open": You can open the motor valve.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Close": You can close the motor valve.
  If the text is configured for this command, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Position Control": You can switch on the position control.
- "Reset": You can reset the motor valve if interlocks or errors occur.
- "Automatic": You can switch to "Automatic mode".
- "Manual": You can switch to "Manual mode".
- "Local": You can switch to "Local mode".
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operating permission": Use the ← button to change to the standard view of the OpStations block. For more information, see section Operator control permissions (Page 285).

### (4) Display current control signals

This area shows the most important parameters for this block with the current selection:

- "Permission" (`Feature2.Bit16` = 0):
  This display is only visible when the corresponding block input is connected.
  - 0 = Motor valve activation not enabled on OS
  - 1 = Enable for "opening"/"closing" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Permission Open" (`Feature2.Bit16` = 1):
  This display is only visible when the corresponding block input is connected.
  - 0 = Motor valve activation not enabled on OS
  - 1 = Enable for "open" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).

- "Permission Close" (`Feature2.Bit16` = 1):
  This display is only visible when the corresponding block input is connected.
  - 0 = Motor valve activation not enabled on OS
  - 1 = Enable for "close" from the neutral position

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Permit neutral pos" (`Feature2.Bit16` = 0):
  This display is only visible when the corresponding block input is connected.
  - 0 = Prevents the control in safe position direction
  - 1 = prevents the stop command

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection" (`Feature2.Bit16` = 0):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection Open" (`Feature2.Bit16` = 1):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Protection Close" (`Feature2.Bit16` = 1):
  This display is only visible when the corresponding block input is connected.
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock" (`Feature2.Bit16` = 0):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).

- "Interlock Open" (`Feature2.Bit16` = 1):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Interlock Close" (`Feature2.Bit16` = 1):
  This display is only visible when the corresponding block input is connected.
  - 0 = Interlocking without reset is active; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = "Good" state

  If the text is configured for the interlocks, it is also displayed in brackets. For more information, see section Labeling of buttons and text (Page 235).
- "Local": 1 = Control signal for "Local mode" (`LocalLi`) is active
- "Motor protection": 1 = Motor is in "good" state
- "Interlock deact.":
  - 0 = Bypass disabled
  - 1 = Bypass interlock in "Local mode" and in simulation
- "Torque opening": 0 = Torque shutoff when opening
- "Torque closing": 0 = Torque shutoff when closing
- "Local stop": 1 = Stop the motor valve in "Local mode"
- "Local open": 1 = Open the motor valve in "Local mode"
- "Local close": 1 = Close the motor valve in "Local mode"
- "Feedback open": 1 = Motor valve is opened
- "Feedback closed": 1 = Motor valve is closed
- "Open": 1 = Motor valve is opening
- "Close": 1 = Motor valve is closing
- "Channel Open": Signal from the output channel block for "Open"
- "Channel Close": Signal from the output channel block for "Close"
- "Feedback opening": 1 = Motor valve is opening
- "Feedback closing": 1 = Motor valve is closing

### (5) Jump key to standard view of any faceplate

This display is only visible when the corresponding block input is interconnected.

Use this navigation button to jump to the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

For more information, see section Opening additional faceplates (Page 234).

### (6) Resynchronization

Displays the Resynchronization state and Resynchronization Signal status based on feedback and readback values.

- When motor feedbacks are not present (`Feature.Bit12` = 1) - Resynchronization preview is based on valve feedbacks.
- When motor feedbacks are present (`Feature.Bit12` = 0) - Resynchronization preview is based on motor feedbacks.

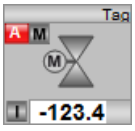
## 8.14.8.6 Block icon for VlvPosL

### Block icons for VlvPosL

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as control system faults
- Operating modes
- Signal status, release for maintenance
- Forcing states
- Operating the manipulated variable in manual mode
- Display of active position control
- Display of the position feedback (white, with decimal places)
- Display for bypassing interlocks
- Interlocks
- Memo display
- Valve status display

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 3 | Non-rotating block icon |
|  | 4 | Non-rotating block icon |

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269)

## Valve status display

The following valve states are shown here:

| Icon | Meaning |
|---|---|
|  | Valve open |
|  | Valve closed |
|  | Error at valve |
|  | Valve is opening |
|  | Valve is closing |
|  | Valve is stopping |
|  | Valve closed |
|  | Valve is closing |

# Interlock blocks

# 9

## 9.1 Intlk02 - Interlock display with 2 input signals

### 9.1.1 Description of Intlk02

#### Object name (type + number) and family

Type + number: FB 1824

Family: Interlck

#### Area of application for Intlk02

The block is used for the following applications:

- Standardized interlock with display

#### How it works

The block is used to calculate a standardized interlock that can be displayed on the OS. A maximum of 2 input signals can be supplied to the block. They are linked using selectable binary logic. The signal status of the output signal is also determined. You can assign an analog value with the signal status and unit to each input value for display in the faceplate.

The current state is displayed at the `Out` output parameter:

- `Out` = 0: Interlock
- `Out` = 1: "Good" state

#### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Intlk02 block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Dosing (Dose_Lean) (Page 2856)
- Dosing with PA/FF devices (Dose_Lean_Fb) (Page 2857)
- Two-speed motor (Motor2Speed) (Page 2860)
- Reversing motor (MotorReversible) (Page 2861)
- Reversible motor with controllable speed (MotorSpeedControlled) (Page 2862)
- Two-way valve (Valve2Way) (Page 2866)

- Motor valve (ValveMotor) (Page 2867)
- Control valve (VlvAnL) (Page 2868)
- Control valve for PA/FF devices (ValveAnalog_Fb) (Page 2869)

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section Intlk02 I/Os (Page 1957).

| Status bit | Parameter |
|---|---|
| 0 | 1 = `Logic` = OR |
| 1 | 1 = `Logic` = AND |
| 2 | Not used |
| 3 | Result of logic operation `Out.Value` |
| 4 | 1 = At least one input value is excluded. See the section Excluding input values in Intlk02 functions (Page 1952). |
| 5 | 1 = All input values are excluded |
| 6 | 1 = No input value is interconnected or `NotUsed.Value` |
| 7 | Status display for simulation |
| 8 | Status display for not interlocked |
| 9 | Status display for interlocked |
| 10 | Activate `OS_Perm` bits |
| 11 | `Feature.Bit2`: Separate evaluation for excluded and simulated interlock signals. |
| 12 | Internal error |
| 13-29 | Not used |
| 30 | `Occupied` |
| 31 | `BatchEn` |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `In01.Value` |
| 1 | `In02.Value` |
| 2 - 31 | Not used |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 | `InvIn01` |
| 1 | `InvIn02` |
| 2 - 31 | Not used |

### Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 | `In01` with inversion |
| 1 | `In02` with inversion |
| 2 - 15 | Not used |
| 16 | Bypass `In01` (via interconnection) |
| 17 | Bypass `In02` (via interconnection) |
| 18 - 31 | Not used |

### Status word allocation for `Status5` parameter

| Status bit | Parameter |
|---|---|
| 0 | `BypIn01` |
| 1 | `BypIn02` |
| 2 - 15 | Not used |
| 16 | In01 hidden bypass information |
| 17 | In02 hidden bypass information |
| 18 - 31 | Not used |

### Status word allocation for `Status6` parameter

| Status bit | Parameter |
|---|---|
| 0 | `In01` not connected |
| 1 | `In02` not connected |
| 2 - 31 | Not used |

### Status word allocation for `Status7` parameter

| Status bit | Parameter |
|---|---|
| 0 | `AV01` not connected |
| 1 | `AV02` not connected |
| 2 - 31 | Not used |

#### Status word allocation for `Status8` parameter

Identical to `FirstIn`.

#### See also

Intlk02 messaging (Page 1957)
Intlk02 block diagram (Page 1961)
Intlk02 error handling (Page 1956)
Intlk02 modes (Page 1952)

### 9.1.2 Intlk02 modes

#### Intlk02 modes

This block does not have any modes.

#### See also

Intlk02 block diagram (Page 1961)
Intlk02 I/Os (Page 1957)
Intlk02 messaging (Page 1957)
Intlk02 error handling (Page 1956)
Intlk02 functions (Page 1952)
Description of Intlk02 (Page 1949)

### 9.1.3 Intlk02 functions

#### Functions of Intlk02

The functions for this block are listed below.

#### Handling non-connected inputs

Inputs which are not interconnected are not evaluated. They are not displayed in the faceplate either.

**Special situation:** If no inputs are connected, the output value is defined using the `DefaultOut` parameter. The signal status is set to 16#FF.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234). However, only one additional faceplate can be called up using `SelFp1 = 1`.

### First-in detection for interlock blocks

This block provides the standard function Recording the first signal for interlock blocks (Page 58). Note that a signal status change only has an effect on the first-in detection when the `Feature` bit Evaluation of signal status (Page 152) and the input `FirstInEn` is set.

The response to deactivation via the `FirstInEn` input can be influenced by `Feature` bit First-in detection response to deactivation (Page 187).

**Note**

This function can only be executed in the faceplate with "process control" operating permission.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for interlock blocks (Page 124).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `Out.ST`

### Bypass

**Note**

Bypassing the interlock means that the interlock signal (input signal) is excluded from the logic of the interlock block, in other words, this signal is ignored in the logic operation.

This function can only be executed in the faceplate with "high-level operating permission".

You can bypass interlock signals as follows:

- From operator over faceplate (`BypInx` = 1):
  Independent `BypLix` or `Inx.Bit1` are connected and enabled, the operator can bypass or reset bypass the interlock input signal.
- By connecting `BypLix` in CFC (`BypLix.Value` = 1, `BypLix.ST` <> `16#FF`)
  Follow this if interlock signals are connected to outputs, which has no bypass information bit. Refer to section Bypassing signals (Page 115).
- By bypass information bit in the connected interlock signal `Inx.Bit1`. For this the corresponding enable bit (`BypEn.Inx` = 1) and `Feature.Bit2` Separate evaluation for excluded and simulated interlock signals (Page 162) are to be set. Follow this if the interlock signals are connected to outputs which has the bypass information bit. Refer to section Bypassing signals (Page 115).

The following table shows the possibilities to switch on/off the bypass of an interlock signal:

| BypLix.ST <> 16#FF | BypEn.Inx AND Feature.Bit2 | Interpretation |
|---|---|---|
| - | - | Only operator: `BypInx` |
| x | - | Input `BypLix` and operator |
| - | x | Bypass bit in Interlock input `Inx.Bit1` and operator |
| x | x | "`BypLix.Value OR Inx.Bit1`" and operator |

A value change of the used bypass signal `(BypLix.Value, Inx.Bit1)` tracks the input `BypInx` for faceplate and actualizes the bypass value. You can always operate the bypass over faceplate if no change of the used connected bypass inputs is active in block cycle.

**Note**

After a change of the parameterization in CFC (bypass via information bit `Bit1` in the interlock signals or bypass via interconnection separate inputs `BypLix`), a corresponding trigger of the bypass value must be made for the change to take effect.

You can also use both connected bypass inputs (`BypLix.Value, Inx.Bit1`) for one interlock input signal. Have a look that all prerequisites are fulfilled. Then with a change of `BypLix.Value` or `Inx.Bit1` the bypass of an interlock input `BypInx` is set to "`BypLix.Value OR Inx.Bit1`". Both inputs are to be false to reset the bypass. After that operator can change bypass over faceplate again.

The following symbol in standard view indicates a bypass request signal from the connected parameter.



- `Inx.Bit1 =1`: Output `Out.Bit1` of an interlock block,
  Output from a technological block with bypass functionality
- `BypLix.Value =1`: Output from a connected block

The following symbol in standard view indicates whether the interlock input signal is in bypass:



At the outputs `Byp01Act`... `Byp02Act`, you can see whether the corresponding input interlock signal is bypassed (`BypxAct` =`BypInx`).

You can reset all the bypass of interlock signals as follows:

- From operator over faceplate (`RstBypOp` = 0 -> 1)
- Over CFC linkable parameter (`RstBypLi` = 0 -> 1)

**Special situation:** If all input parameters are bypassed, the output value is defined using the `DefaultOut` parameter.

The output BypAct displays the information if there is an operated or connected bypass available.

```
BypAct.Value = BypIn01 OR BypInx …
                       OR In01.Bit1 OR Inx.Bit1 …
                       OR BypLi1.Value OR BypLix.Value …
```

**Note**

Do not connect the input BypLix with the output BypAct from an interlock block as BypAct is calculated from the internal bypass signal information and is only for display bypass and not for operate a bypass of an interlock input.

The bypass information Out.Bit1 and the status Out.ST depends on Feature.Bit2:

**Feature.Bit2 = 0:**

Out.Bit1 = 0, Bypass information bit is not active and reset.
A bypassed interlock input, which is switch-relevant, sets the status of Out.ST to Simulation.

**Feature.Bit2 = 1:**

Out.Bit1 = BypAct.Value, Bypass information bit is active and tracked.
A bypassed interlock input has no influence on the status of Out.ST

## Operating permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the OS_Perm parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can exclude values |
| 1 | 1 = Operator can reset the exclusion of input values |
| 2 | 1 = Operator can reset the recording of the first signal |
| 3 - 15 | Not used |
| 16 | 1 = Operator can set or reset the exclusion of input value In01 |
| 17 | 1 = Operator can set or reset the exclusion of input value In02 |
| 18 – 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in OS_Perm as a parameter, you have to reset the corresponding OS_Perm bit.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the chapter Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 5 | Activate OS_Perm bits (Page 168)<br>0 = `OS_Perm` bits 16...17 inactive (evaluation only in faceplate)<br>1 = `OS_Perm` bits 16...17 active (evaluation only in faceplate) |
| 21 | First-in detection response to deactivation (Page 187) |
| 23 | Evaluation of signal status (Page 152) |
| 24 | Enabling local operator authorization (Page 169) |
| 31 | Activating recording of the first signal (Page 161) |

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of Intlk02 (Page 1949)

Intlk02 messaging (Page 1957)

Intlk02 I/Os (Page 1957)

Intlk02 block diagram (Page 1961)

Intlk02 error handling (Page 1956)

Intlk02 modes (Page 1952)

## 9.1.4 Intlk02 error handling

### Error handling of Intlk02

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 13 | The parameter `Logic` has not been set to 0 or 1. |

### See also

Intlk02 block diagram (Page 1961)
Intlk02 I/Os (Page 1957)
Intlk02 messaging (Page 1957)
Intlk02 functions (Page 1952)
Intlk02 modes (Page 1952)
Description of Intlk02 (Page 1949)

## 9.1.5 Intlk02 messaging

### Messaging

This block does not offer messaging.

### See also

Description of Intlk02 (Page 1949)
Intlk02 functions (Page 1952)
Intlk02 I/Os (Page 1957)
Intlk02 block diagram (Page 1961)
Intlk02 error handling (Page 1956)
Intlk02 modes (Page 1952)

## 9.1.6 Intlk02 I/Os

I/Os of Intlk02

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

## Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AV01 | Analog value of In01 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV02 | Analog value of In02 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV01_Unit | Unit of measure for AV01 | INT | 0 |
| AV02_Unit | Unit of measure for AV02 | INT | 0 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BypIn01* | 1 = Input In01 is not used | BOOL | 0 |
| BypIn02* | 1 = Input In02 is not used | BOOL | 0 |
| BypLi01 | Bypass In01 (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| BypLi02 | Bypass In02 (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| DefaultOut | Output value for cases where all inputs are excluded or not interconnected. Refer to the section Intlk02 functions (Page 1952) of the block. | BOOL | 1 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 1952) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| FirstInEn | First-in detection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| In01 | Input In01 | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| In02 | Input In02 | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| InvIn01 | Invert input In01 | BOOL | 0 |
| InvIn02 | Invert input In02 | BOOL | 0 |
| Logic | Logical operation:<br>0 = Logical OR<br>1 = Logical AND | INT | 0 |
| NotUsed | 1 = Block is not used (only for display in the faceplate). You can find additional information on this in the section Interlock blocks standard view (Page 310) / Color of the field. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OS_Perm | I/O for operating permissions (Page 1952) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| RstBypOp* | 1 = Reset via BypIn01and BypIn02 | BOOL | 0 |
| RstLi | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | 1 = Reset via operator | BOOL | 0 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | |
| StepNo | Batch step number | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Data type | Default |
|---|---|---|---|
| BypAct | 1 = Bypass active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Byp01Act | 1 = Bypass of input 01 is active | BOOL | 0 |
| Byp02Act | 1 = Bypass of input 02 is active | BOOL | 0 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see Intlk02 error handling (Page 1956). | INT | -1 |
| FirstIn | Bit-coded number of the first signal that has resulted in a change at the output | DWORD | 16#00000000 |
| OpSt_Out | Value of the OpSt_In input parameter for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| Out | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 1949) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1949) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1949) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 1949) | DWORD | 16#00000000 |
| Status5 | Status word 5 (Page 1949) | DWORD | 16#00000000 |
| Status6 | Status word 6 (Page 1949) | DWORD | 16#00000000 |
| Status7 | Status word 7 (Page 1949) | DWORD | 16#00000000 |
| Status8 | Status word 8 (Page 1949) | DWORD | 16#00000000 |

**See also**

Intlk02 messaging (Page 1957)

Intlk02 block diagram (Page 1961)

Intlk02 modes (Page 1952)

## 9.1.7 Intlk02 block diagram

### Intlk02 block diagram

A block diagram is not provided for this block.

**See also**

Intlk02 I/Os (Page 1957)

Intlk02 messaging (Page 1957)

Intlk02 error handling (Page 1956)

Intlk02 functions (Page 1952)

Intlk02 modes (Page 1952)

Description of Intlk02 (Page 1949)

## 9.1.8 Operator control and monitoring

### 9.1.8.1 Interlock block views

### Views of the blocks Intlk02, Intlk04, Intlk08, Intlk16

The blocks provide the following views:

- Interlock blocks standard view (Page 310)
- Preview of interlock blocks (Page 333)
- Batch view (Page 336)
- Block icon for interlock blocks (Page 272)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

# 9.2 Intlk04 - Interlock display with 4 input signals

## 9.2.1 Description of Intlk04

### Object name (type + number) and family

Type + number: FB 1825

Family: Interlck

### Area of application for Intlk04

The block is used for the following applications:

- Standardized interlock with display

### How it works

The block is used to calculate a standardized interlock that can be displayed on the OS. A maximum of 4 input signals can be supplied to the block. They are linked using selectable binary logic. The signal status of the output signal is also determined. You can assign an analog value with the signal status and unit to each input value for display in the faceplate.

The current state is displayed at the `Out` output parameter:

- `Out` = 0: Interlock
- `Out` = 1: "Good" state

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Intlk02 (Intlk04) block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Refer to Description of Intlk02 (Page 1949) for more information.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section Intlk04 I/Os (Page 1970).

| Status bit | Parameter |
|---|---|
| 0 | 1= `Logic` = OR |
| 1 | 1 = `Logic` = AND |
| 2 | Not used |

| Status bit | Parameter |
|---|---|
| 3 | Result of logic operation `Out.Value` |
| 4 | 1 = At least one input value is excluded. See the section Excluding input values in Intlk04 functions (Page 1965). |
| 5 | 1 = All input values are excluded |
| 6 | 1 = No input value is interconnected or `NotUsed.Value` |
| 7 | Status display for simulation |
| 8 | Status display for not interlocked |
| 9 | Status display for interlocked |
| 10 | Not used |
| 11 | `Feature.Bit2`: Separate evaluation for excluded and simulated interlock signals |
| 12 | Internal error |
| 13-29 | Not used |
| 30 | `Occupied` |
| 31 | `BatchEn` |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `In01.Value` |
| 1 | `In02.Value` |
| 2 | `In03.Value` |
| 3 | `In04.Value` |
| 4 - 31 | Not used |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 | `InvIn01` |
| 1 | `InvIn02` |
| 2 | `InvIn03` |
| 3 | `InvIn04` |
| 4 - 31 | Not used |

### Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 | `In01` with inversion |
| 1 | `In02` with inversion |
| 2 | `In03` with inversion |
| 3 | `In04` with inversion |
| 4 - 15 | Not used |
| 16 | Bypass `In01` (via interconnection) |

| Status bit | Parameter |
|---|---|
| 17 | Bypass `In02` (via interconnection) |
| 18 | Bypass `In03` (via interconnection) |
| 19 | Bypass `In04` (via interconnection) |
| 20 - 31 | Not used |

### Status word allocation for `Status5` parameter

| Status bit | Parameter |
|---|---|
| 0 | `BypIn01` |
| 1 | `BypIn02` |
| 2 | `BypIn03` |
| 3 | `BypIn04` |
| 4 - 15 | Not used |
| 16 | In01hidden bypass information |
| 17 | In02hidden bypass information |
| 18 | In03hidden bypass information |
| 19 | In04hidden bypass information |
| 20-31 | Not used |

### Status word allocation for `Status6` parameter

| Status bit | Parameter |
|---|---|
| 0 | `In01` not connected |
| 1 | `In02` not connected |
| 2 | `In03` not connected |
| 3 | `In04` not connected |
| 4 - 31 | Not used |

### Status word allocation for `Status7` parameter

| Status bit | Parameter |
|---|---|
| 0 | `AV01` not connected |
| 1 | `AV02` not connected |
| 2 | `AV03` not connected |
| 3 | `AV04` not connected |
| 4 - 31 | Not used |

### Status word allocation for `Status8` parameter

Identical to `FirstIn`.

**See also**

Intlk04 messaging (Page 1970)
Intlk04 block diagram (Page 1974)
Intlk04 error handling (Page 1969)
Intlk04 modes (Page 1965)

## 9.2.2 Intlk04 modes

### Intlk04 modes

This block does not have any modes.

**See also**

Intlk04 block diagram (Page 1974)
Intlk04 I/Os (Page 1970)
Intlk04 messaging (Page 1970)
Intlk04 error handling (Page 1969)
Intlk04 functions (Page 1965)
Description of Intlk04 (Page 1962)

## 9.2.3 Intlk04 functions

### Functions of Intlk04

The functions for this block are listed below.

### Inversion of logic signals

You can invert the input signals by setting the input parameter `InvInx` for the input concerned to `Inx` = 1, e.g. for input `In01` set the I/O `InvIn01`.

The inversion is displayed in the faceplate. If you invert signals using any other method this is not shown in the faceplate.

## Bypass

**Note**

Bypassing the interlock means that the interlock signal (input signal) is excluded from the logic of the interlock block, in other words, this signal is ignored in the logic operation.

This function can only be executed in the faceplate with "high-level operating permission".

You can bypass interlock signals as follows:

- From operator over faceplate (`BypInx` = 1):
  Independent `BypLix` or `Inx.Bit1` are connected and enabled, the operator can bypass or reset bypass the interlock input signal.
- By connecting `BypLix` in CFC (`BypLix.Value` = 1, `BypLix.ST` <> `16#FF`)
  Follow this if interlock signals are connected to outputs, which has no bypass information bit. Refer to sectionBypassing signals (Page 115).
- By bypass information bit in the connected interlock signal `Inx.Bit1`. For this the corresponding enable bit (`BypEn.Inx = 1`) and `Feature.Bit2` Separate evaluation for excluded and simulated interlock signals (Page 162) are to be set. Follow this if the interlock signals are connected to outputs which has the bypass information bit. Refer to sectionBypassing signals (Page 115).

The following table shows the possibilities to switch on/off the bypass of an interlock signal:

| BypLix.ST <> 16#FF | BypEn.Inx AND Feature.Bit2 | Interpretation |
|---|---|---|
| - | - | Only operator: `BypInx` |
| x | - | Input `BypLix` and operator |
| - | x | Bypass bit in Interlock input `Inx.Bit1` and operator |
| x | x | "`BypLix.Value OR Inx.Bit1`" and operator |

A value change of the used bypass signal `(BypLix.Value, Inx.Bit1)` tracks the input `BypInx` for faceplate and actualizes the bypass value. You can always operate the bypass over faceplate if no change of the used connected bypass inputs is active in block cycle.

**Note**

After a change of the parameterization in CFC (bypass via information bit `Bit1` in the interlock signals or bypass via interconnection separate inputs `BypLix`), a corresponding trigger of the bypass value must be made for the change to take effect.

You can also use both connected bypass inputs (`BypLix.Value, Inx.Bit1`) for one interlock input signal. Have a look that all prerequisites are fulfilled. Then with a change of `BypLix.Value` or `Inx.Bit1` the bypass of an interlock input `BypInx` is set to "`BypLix.Value OR Inx.Bit1`". Both inputs are to be false to reset the bypass. After that operator can change bypass over faceplate again.

The following symbol in standard view indicates a bypass request signal from the connected parameter.



- `Inx.Bit1 =1`: Output `Out.Bit1` of an interlock block,
  Output from a technological block with bypass functionality
- `BypLix.Value =1`: Output from a connected block

The following symbol in standard view indicates whether the interlock input signal is in bypass:



At the outputs `Byp01Act`... `Byp02Act`, you can see whether the corresponding input interlock signal is bypassed (`BypxAct` =`BypInx`).

You can reset all the bypass of interlock signals as follows:

- From operator over faceplate (`RstBypOp` = 0 -> 1)
- Over CFC linkable parameter (`RstBypLi` = 0 -> 1)

**Special situation:** If all input parameters are bypassed, the output value is defined using the `DefaultOut` parameter.

The output `BypAct` displays the information if there is an operated or connected bypass available.

```
BypAct.Value = BypIn01 OR BypInx …
                       OR In01.Bit1 OR Inx.Bit1 …
                       OR BypLi1.Value OR BypLix.Value …
```

**Note**

Do not connect the input `BypLix` with the output `BypAct` from an interlock block as `BypAct` is calculated from the internal bypass signal information and is only for display bypass and not for operate a bypass of an interlock input.

The bypass information `Out.Bit1` and the status `Out.ST` depends on `Feature.Bit2`:

**`Feature.Bit2` = 0:**

`Out.Bit1` = 0, Bypass information bit is not active and reset.
A bypassed interlock input, which is switch-relevant, sets the status of `Out.ST` to Simulation.

**`Feature.Bit2` = 1:**

`Out.Bit1` = `BypAct.Value`, Bypass information bit is active and tracked.
A bypassed interlock input has no influence on the status of `Out.ST`

## Handling non-connected inputs

Inputs which are not interconnected are not evaluated. They are not displayed in the faceplate either.

**Special situation:** If no inputs are connected, the output value is defined using the `DefaultOut` parameter. The signal status is set to 16#FF.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234). However, only one additional faceplate can be called up using `SelFp1` = 1.

### First-in detection for interlock blocks

This block provides the standard function Recording the first signal for interlock blocks (Page 58). Note that a signal status change only has an effect on the first-in detection when the `Feature` bit Evaluation of signal status (Page 152) and the input `FirstInEn` is set.

The response to deactivation via the `FirstInEn` input can be influenced by `Feature` bit First-in detection response to deactivation (Page 187).

**Note**

This function can only be executed in the faceplate with "process control" operating permission.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for interlock blocks (Page 124).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `Out.ST`

### Operating permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can exclude values |
| 1 | 1 = Operator can reset the exclusion of input values |
| 2 | 1 = Operator can reset the recording of the first signal |
| 3 - 15 | Not used |
| 16 | 1 = Operator can set or reset the exclusion of input value In01 |
| 17 | 1 = Operator can set or reset the exclusion of input value In02 |
| 18 | 1 = Operator can set or reset the exclusion of input value In03 |
| 19 | 1 = Operator can set or reset the exclusion of input value In04 |
| 20 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Configurable reactions using the `Feature` I/O

You can find an overview of all reactions provided by the `Feature` parameter in the chapter Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 5 | Activate OS_Perm bits (Page 168)<br>0 = `OS_Perm` bits 16...17 inactive (evaluation only in faceplate)<br>1 = `OS_Perm` bits 16...17 active (evaluation only in faceplate) |
| 21 | First-in detection response to deactivation (Page 187) |
| 23 | Evaluation of signal status (Page 152) |
| 24 | Enabling local operator authorization (Page 169) |
| 31 | Activating recording of the first signal (Page 161) |

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of Intlk04 (Page 1962)

Intlk04 messaging (Page 1970)

Intlk04 I/Os (Page 1970)

Intlk04 block diagram (Page 1974)

Intlk04 error handling (Page 1969)

Intlk04 modes (Page 1965)

## 9.2.4 Intlk04 error handling

### Error handling of Intlk04

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 13 | The parameter `Logic` has not been set to 0 or 1. |

### See also

Intlk04 block diagram (Page 1974)
Intlk04 I/Os (Page 1970)
Intlk04 messaging (Page 1970)
Description of Intlk04 (Page 1962)
Intlk04 modes (Page 1965)
Intlk04 functions (Page 1965)

## 9.2.5 Intlk04 messaging

### Messaging

This block does not offer messaging.

### See also

Description of Intlk04 (Page 1962)
Intlk04 functions (Page 1965)
Intlk04 I/Os (Page 1970)
Intlk04 block diagram (Page 1974)
Intlk04 error handling (Page 1969)
Intlk04 modes (Page 1965)

## 9.2.6 Intlk04 I/Os

I/Os of Intlk04

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

## Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AV01 | Analog value of In01 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV02 | Analog value of In02 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV03 | Analog value of In03 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV04 | Analog value of In04 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV01_Unit | Unit of measure for AV01 | INT | 0 |
| AV02_Unit | Unit of measure for AV02 | INT | 0 |
| AV03_Unit | Unit of measure for AV03 | INT | 0 |
| AV04_Unit | Unit of measure for AV04 | INT | 0 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BypIn01* | 1 = Input In01 is not used | BOOL | 0 |
| BypIn02* | 1 = Input In02 is not used | BOOL | 0 |
| BypIn03* | 1 = Input In03 is not used | BOOL | 0 |
| BypIn04* | 1 = Input In04 is not used | BOOL | 0 |
| BypLi01...BypLi04 | Bypass In01...In04 (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| DefaultOut | Output value for the case that all inputs are excluded or not connected. Refer to section Intlk04 functions (Page 1965) of the block. | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 1965) | STRUCT | - |
| | | • Bit 0: BOOL | • 0 |
| | | • ... | • 0 |
| | | • Bit 31: BOOL | • 0 |
| FirstInEn | First-in detection | STRUCT | - |
| | | • Value: BOOL | • 1 |
| | | • ST: BYTE | • 16#80 |
| In01 | Input In01 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#FF |
| | | • Bit 1: BOOL (Bypass information bit) | • 0 |
| In02 | Input In02 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#FF |
| | | • Bit 1: BOOL (Bypass information bit) | • 0 |
| In03 | Input In03 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#FF |
| | | • Bit 1: BOOL (Bypass information bit) | • 0 |
| In04 | Input In04 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#FF |
| | | • Bit 1: BOOL (Bypass information bit) | • 0 |
| InvIn01 | Invert input In01 | BOOL | 0 |
| InvIn02 | Invert input In02 | BOOL | 0 |
| InvIn03 | Invert input In03 | BOOL | 0 |
| InvIn04 | Invert input In04 | BOOL | 0 |
| Logic | Logical operation:<br>0 = Logical OR<br>1 = Logical AND | INT | 0 |
| NotUsed | 1 = Block is not used (only for display in the faceplate). You can find additional information on this in the section Interlock blocks standard view (Page 310) / Color of the field. | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operating permissions (Page 1965) | STRUCT<br>• Bit 0: BOOL<br>• ....<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| RstBypOp* | 1 = Reset inputs BypIn01 to BypIn04 | BOOL | 0 |
| RstLi | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | 1 = Reset via operator | BOOL | 0 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| BypAct | 1 = Bypass active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Byp01Act | 1 = Bypass of input 01 is active | BOOL | 0 |
| Byp02Act | 1 = Bypass of input 02 is active | BOOL | 0 |
| Byp03Act | 1 = Bypass of input 03 is active | BOOL | 0 |
| Byp04Act | 1 = Bypass of input 04 is active | BOOL | 0 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see Intlk04 error handling (Page 1969) | INT | -1 |
| FirstIn | Bit-coded number of the first signal that has resulted in a change at the output | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| Out | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 1962) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1962) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1962) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 1962) | DWORD | 16#00000000 |
| Status5 | Status word 5 (Page 1962) | DWORD | 16#00000000 |
| Status6 | Status word 6 (Page 1962) | DWORD | 16#00000000 |
| Status7 | Status word 7 (Page 1962) | DWORD | 16#00000000 |
| Status8 | Status word 8 (Page 1962) | DWORD | 16#00000000 |

### See also

Intlk04 messaging (Page 1970)

Intlk04 block diagram (Page 1974)

Intlk04 modes (Page 1965)

## 9.2.7 Intlk04 block diagram

### Intlk04 block diagram

A block diagram is not provided for this block.

### See also

Intlk04 I/Os (Page 1970)

Intlk04 messaging (Page 1970)

Intlk04 error handling (Page 1969)

Intlk04 functions (Page 1965)

Intlk04 modes (Page 1965)

Description of Intlk04 (Page 1962)

## 9.2.8 Operator control and monitoring

### 9.2.8.1 Interlock block views

#### Views of the blocks Intlk02, Intlk04, Intlk08, Intlk16

The blocks provide the following views:

- Interlock blocks standard view (Page 310)
- Preview of interlock blocks (Page 333)
- Batch view (Page 336)
- Block icon for interlock blocks (Page 272)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

# 9.3 Intlk08 - Interlock display with 8 input signals

## 9.3.1 Description of Intlk08

### Object name (type + number) and family

Type + number: FB 1826

Family: Interlck

### Area of application of Intlk08

The block is used for the following applications:

- Standardized interlock with display

### How it works

The block is used to calculate a standardized interlock that can be displayed on the OS. A maximum of 8 input signals can be supplied to the block. They are linked using selectable binary logic. The signal status of the output signal is also determined. You can assign an analog value with the signal status and unit to each input value for display in the faceplate.

The current state is displayed at the `Out` output parameter:

- `Out` = 0: Interlock
- `Out` = 1: "Good" state

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Intlk02 (Intlk08) block, the Advanced Process Library contains process tag type templates; these serve as examples by providing various application scenarios for this block.

Refer to Description of Intlk02 (Page 1949) for more information.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section Intlk08 I/Os (Page 1985).

| Status bit | Parameter |
|---|---|
| 0 | 1 = `Logic` = OR |
| 1 | 1 = `Logic` = AND |
| 2 | Not used |

| Status bit | Parameter |
|---|---|
| 3 | Result of logic operation `Out.Value` |
| 4 | 1 = At least one input value is excluded. See the section Excluding input values in Intlk08 functions (Page 1980) |
| 5 | 1 = All input values are excluded |
| 6 | 1 = No input value is interconnected or `NotUsed.Value` |
| 7 | Status display for simulation |
| 8 | Status display for not interlocked |
| 9 | Status display for interlocked |
| 10 | `Feature.Bit5` |
| 11 | `Feature.Bit2`: Separate evaluation for excluded and simulated interlock signals |
| 12 | Internal error |
| 13-29 | Not used |
| 30 | `Occupied` |
| 31 | `BatchEn` |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `In01.Value` |
| 1 | `In02.Value` |
| 2 | `In03.Value` |
| 3 | `In04.Value` |
| 4 | `In05.Value` |
| 5 | `In06.Value` |
| 6 | `In07.Value` |
| 7 | `In08.Value` |
| 8 - 31 | Not used |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 | `InvIn01` |
| 1 | `InvIn02` |
| 2 | `InvIn03` |
| 3 | `InvIn04` |
| 4 | `InvIn05` |
| 5 | `InvIn06` |
| 6 | `InvIn07` |
| 7 | `InvIn08` |
| 8 - 31 | Not used |

### Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 | `In01` with inversion |
| 1 | `In02` with inversion |
| 2 | `In03` with inversion |
| 3 | `In04` with inversion |
| 4 | `In05` with inversion |
| 5 | `In06` with inversion |
| 6 | `In07` with inversion |
| 7 | `In08` with inversion |
| 8 - 15 | Not used |
| 16 | Bypass `In01` (via interconnection) |
| 17 | Bypass `In02` (via interconnection) |
| 18 | Bypass `In03` (via interconnection) |
| 19 | Bypass `In04` (via interconnection) |
| 20 | Bypass `In05` (via interconnection) |
| 21 | Bypass `In06` (via interconnection) |
| 22 | Bypass `In07` (via interconnection) |
| 23 | Bypass `In08` (via interconnection) |
| 24 - 31 | Not used |

### Status word allocation for `Status5` parameter

| Status bit | Parameter |
|---|---|
| 0 | `BypIn01` |
| 1 | `BypIn02` |
| 2 | `BypIn03` |
| 3 | `BypIn04` |
| 4 | `BypIn05` |
| 5 | `BypIn06` |
| 6 | `BypIn07` |
| 7 | `BypIn08` |
| 8 - 15 | Not used |
| 16 - 23 | `In01..IN08` hidden bypass information |
| 24 - 31 | Not used |

### Status word allocation for `Status6` parameter

| Status bit | Parameter |
|---|---|
| 0 | `In01` not connected |
| 1 | `In02` not connected |
| 2 | `In03` not connected |

| Status bit | Parameter |
|---|---|
| 3 | `In04` not connected |
| 4 | `In05` not connected |
| 5 | `In06` not connected |
| 6 | `In07` not connected |
| 7 | `In08` not connected |
| 8 - 31 | Not used |

### Status word allocation for `Status7` parameter

| Status bit | Parameter |
|---|---|
| 0 | `AV01` not connected |
| 1 | `AV02` not connected |
| 2 | `AV03` not connected |
| 3 | `AV04` not connected |
| 4 | `AV05` not connected |
| 5 | `AV06` not connected |
| 6 | `AV07` not connected |
| 7 | `AV08` not connected |
| 8 - 31 | Not used |

### Status word allocation for `Status8` parameter

Identical to `FirstIn`.

### See also

Intlk08 messaging (Page 1985)
Intlk08 block diagram (Page 1991)
Intlk08 error handling (Page 1984)
Intlk08 modes (Page 1979)

## 9.3.2 Intlk08 modes

### Intlk08 modes

This block does not have any modes.

### See also

Intlk08 block diagram (Page 1991)
Intlk08 I/Os (Page 1985)

Intlk08 messaging (Page 1985)
Intlk08 error handling (Page 1984)
Intlk08 functions (Page 1980)
Description of Intlk08 (Page 1976)

## 9.3.3 Intlk08 functions

### Functions of Intlk08

The functions for this block are listed below.

### Logic operators

Use the `Logic` input to specify the logic operator that the block should employ when determining the interlock state. Make the following settings:

- `Logic` = 0: OR
- `Logic` = 1: AND

### Inversion of logic signals

You can invert the input signals by setting the input parameter `InvInx` for the input concerned to `Inx` = 1 , e.g. for input `In01` set the I/O `InvIn01`.

The inversion is displayed in the faceplate. If you invert signals using any other method, this is not shown in the faceplate.

### Bypass

**Note**

Bypassing the interlock means that the interlock signal (input signal) is excluded from the logic of the interlock block, in other words, this signal is ignored in the logic operation.

This function can only be executed in the faceplate with "high-level operating permission".

You can bypass interlock signals as follows:

- From operator over faceplate (`BypInx` = 1):
  Independent `BypLix` or `Inx.Bit1` are connected and enabled, the operator can bypass or reset bypass the interlock input signal.
- By connecting `BypLix` in CFC (`BypLix.Value` = 1, `BypLix.ST` <> `16#FF`)
  Follow this if interlock signals are connected to outputs, which has no bypass information bit. Refer to sectionBypassing signals (Page 115).
- By bypass information bit in the connected interlock signal `Inx.Bit1`. For this the corresponding enable bit (`BypEn.Inx` = 1) and `Feature.Bit2` Separate evaluation for excluded and simulated interlock signals (Page 162) are to be set. Follow this if the interlock signals are connected to outputs which has the bypass information bit. Refer to sectionBypassing signals (Page 115).

The following table shows the possibilities to switch on/off the bypass of an interlock signal:

| BypLix.ST <> 16#FF | BypEn.Inx AND Feature.Bit2 | Interpretation |
|---|---|---|
| - | - | Only operator: `BypInx` |
| x | - | Input `BypLix` and operator |
| - | x | Bypass bit in Interlock input `Inx.Bit1` and operator |
| x | x | "`BypLix.Value OR Inx.Bit1`" and operator |

A value change of the used bypass signal `(BypLix.Value, Inx.Bit1)` tracks the input `BypInx` for faceplate and actualizes the bypass value. You can always operate the bypass over faceplate if no change of the used connected bypass inputs is active in block cycle.

**Note**

After a change of the parameterization in CFC (bypass via information bit `Bit1` in the interlock signals or bypass via interconnection separate inputs `BypLix`), a corresponding trigger of the bypass value must be made for the change to take effect.

You can also use both connected bypass inputs (`BypLix.Value, Inx.Bit1`) for one interlock input signal. Have a look that all prerequisites are fulfilled. Then with a change of `BypLix.Value` or `Inx.Bit1` the bypass of an interlock input `BypInx` is set to "`BypLix.Value OR Inx.Bit1`". Both inputs are to be false to reset the bypass. After that operator can change bypass over faceplate again.

The following symbol in standard view indicates a bypass request signal from the connected parameter.



- `Inx.Bit1 =1`: Output `Out.Bit1` of an interlock block,
  Output from a technological block with bypass functionality
- `BypLix.Value =1`: Output from a connected block

The following symbol in standard view indicates whether the interlock input signal is in bypass:



At the outputs `Byp01Act`... `Byp02Act`, you can see whether the corresponding input interlock signal is bypassed (`BypxAct` =`BypInx`).

You can reset all the bypass of interlock signals as follows:

- From operator over faceplate (`RstBypOp` = 0 -> 1)
- Over CFC linkable parameter (`RstBypLi` = 0 -> 1)

**Special situation:** If all input parameters are bypassed, the output value is defined using the `DefaultOut` parameter.

The output `BypAct` displays the information if there is an operated or connected bypass available.

```
BypAct.Value = BypIn01 OR BypInx …
                        OR In01.Bit1 OR Inx.Bit1 …
                        OR BypLi1.Value OR BypLix.Value …
```

**Note**

Do not connect the input `BypLix` with the output `BypAct` from an interlock block as `BypAct` is calculated from the internal bypass signal information and is only for display bypass and not for operate a bypass of an interlock input.

The bypass information `Out.Bit1` and the status `Out.ST` depends on `Feature.Bit2`:

**`Feature.Bit2` = 0:**

`Out.Bit1` = 0, Bypass information bit is not active and reset.
A bypassed interlock input, which is switch-relevant, sets the status of `Out.ST` to Simulation.

**`Feature.Bit2` = 1:**

`Out.Bit1` = `BypAct.Value`, Bypass information bit is active and tracked.
A bypassed interlock input has no influence on the status of `Out.ST`

## Handling non-connected inputs

Inputs which are not interconnected are not evaluated. They are not displayed in the faceplate either.

**Special situation:** If no inputs are connected, the output value is defined using the `DefaultOut` parameter. The signal status is set to 16#FF.

## Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234). However, only one additional faceplate can be called up using `SelFp1` = 1 .

### First-in detection for interlock blocks

This block provides the standard function Recording the first signal for interlock blocks (Page 58). Note that a signal status change only has an effect on the first-in detection when the `Feature` bit Evaluation of signal status (Page 152) and the input FirstInEn is set.

The response to deactivation via the `FirstInEn` input can be influenced by `Feature` bit First-in detection response to deactivation (Page 187).

**Note**

This function can only be executed in the faceplate with "process control" operating permission.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for interlock blocks (Page 124).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `Out.ST`

### Operating permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can exclude values |
| 1 | 1 = Operator can reset the exclusion of input values |
| 2 | 1 = Operator can reset the recording of the first signal |
| 3 - 15 | Not used |
| 16 | 1 = Operator can set or reset the exclusion of input value In01 |
| 17 | 1 = Operator can set or reset the exclusion of input value In02 |
| 18 | 1 = Operator can set or reset the exclusion of input value In03 |
| 19 | 1 = Operator can set or reset the exclusion of input value In04 |
| 20 | 1 = Operator can set or reset the exclusion of input value In05 |
| 21 | 1 = Operator can set or reset the exclusion of input value In06 |
| 22 | 1 = Operator can set or reset the exclusion of input value In07 |
| 23 | 1 = Operator can set or reset the exclusion of input value In08 |
| 24 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the chapter Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 5 | Activate OS_Perm bits (Page 168)<br>0 = `OS_Perm` bits 16...17 inactive (evaluation only in faceplate)<br>1 = `OS_Perm` bits 16...17 active (evaluation only in faceplate) |
| 21 | First-in detection response to deactivation (Page 187) |
| 23 | Evaluation of signal status (Page 152) |
| 24 | Enabling local operator authorization (Page 169) |
| 31 | Activating recording of the first signal (Page 161) |

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of Intlk08 (Page 1976)

Intlk08 messaging (Page 1985)

Intlk08 I/Os (Page 1985)

Intlk08 block diagram (Page 1991)

Intlk08 error handling (Page 1984)

Intlk08 modes (Page 1979)

## 9.3.4 Intlk08 error handling

### Error handling of Intlk08

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 13 | The parameter `Logic` has not been set to 0 or 1 |

### See also

Intlk08 block diagram (Page 1991)
Intlk08 I/Os (Page 1985)
Intlk08 messaging (Page 1985)
Intlk08 functions (Page 1980)
Intlk08 modes (Page 1979)
Description of Intlk08 (Page 1976)

## 9.3.5 Intlk08 messaging

### Messaging

This block does not offer messaging.

### See also

Description of Intlk08 (Page 1976)
Intlk08 functions (Page 1980)
Intlk08 I/Os (Page 1985)
Intlk08 block diagram (Page 1991)
Intlk08 error handling (Page 1984)
Intlk08 modes (Page 1979)

## 9.3.6 Intlk08 I/Os

I/Os of Intlk08

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

## Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AV01 | Analog value of In01 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV02 | Analog value of In02 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV03 | Analog value of In03 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV04 | Analog value of In04 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV05 | Analog value of In05 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV06 | Analog value of In06 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV07 | Analog value of In07 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV08 | Analog value of In08 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV01_Unit | Unit of measure for AV01 | INT | 0 |
| AV02_Unit | Unit of measure for AV02 | INT | 0 |
| AV03_Unit | Unit of measure for AV03 | INT | 0 |
| AV04_Unit | Unit of measure for AV04 | INT | 0 |
| AV05_Unit | Unit of measure for AV05 | INT | 0 |
| AV06_Unit | Unit of measure for AV06 | INT | 0 |
| AV07_Unit | Unit of measure for AV07 | INT | 0 |
| AV08_Unit | Unit of measure for AV08 | INT | 0 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| BlkTypeInfo* | Block Type Version and FB Number of block-type<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BypIn01* | 1 = Input In01 is not used | BOOL | 0 |
| BypIn02* | 1 = Input In02 is not used | BOOL | 0 |
| BypIn03* | 1 = Input In03 is not used | BOOL | 0 |
| BypIn04* | 1 = Input In04 is not used | BOOL | 0 |
| BypIn05* | 1 = Input In05 is not used | BOOL | 0 |
| BypIn06* | 1 = Input In06 is not used | BOOL | 0 |
| BypIn07* | 1 = Input In07 is not used | BOOL | 0 |
| BypIn08* | 1 = Input In08 is not used | BOOL | 0 |
| BypLi01...BypLi08 | Bypass In01...In08 (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| DefaultOut | Output value for cases where all inputs are excluded or not interconnected. Refer to the section Intlk08 functions (Page 1980) of the block. | BOOL | 1 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 1980) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| FirstInEn | First-in detection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| In01 | Input In01 | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| `In02` | Input `In02` | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| `In03` | Input `In03` | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| `In04` | Input `In04` | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| `In05` | Input `In05` | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| `In06` | Input In06 | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| `In07` | Input `In07` | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| `In08` | Input `In08` | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| `InvIn01` | Invert Input `In01` | BOOL | 0 |
| `InvIn02` | Invert Input `In02` | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| InvIn03 | Invert Input In03 | BOOL | 0 |
| InvIn04 | Invert Input In04 | BOOL | 0 |
| InvIn05 | Invert Input In05 | BOOL | 0 |
| InvIn06 | Invert Input In06 | BOOL | 0 |
| InvIn07 | Invert Input In07 | BOOL | 0 |
| InvIn08 | Invert Input In08 | BOOL | 0 |
| Logic | Logical operation:<br>0 = Logical OR<br>1 = Logical AND | INT | 0 |
| NotUsed | 1 = Block is not used (only for display in the faceplate). You can find additional information on this in the section Interlock blocks standard view (Page 310) / Color of the field. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operating permissions (Page 1980) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| RstBypOp* | 1 = Reset inputs BypIn01to BypIn08 | BOOL | 0 |
| RstLi | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | 1 = Reset via operator | BOOL | 0 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| BypAct | 1 = Bypass active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Byp01Act | 1 = Bypass of input 01 is active | BOOL | 0 |
| Byp02Act | 1 = Bypass of input 02 is active | BOOL | 0 |
| Byp03Act | 1 = Bypass of input 03 is active | BOOL | 0 |
| Byp04Act | 1 = Bypass of input 04 is active | BOOL | 0 |
| Byp05Act | 1 = Bypass of input 05 is active | BOOL | 0 |
| Byp06Act | 1 = Bypass of input 06 is active | BOOL | 0 |
| Byp07Act | 1 = Bypass of input 07 is active | BOOL | 0 |
| Byp08Act | 1 = Bypass of input 08 is active | BOOL | 0 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see Intlk08 error handling (Page 1984) | INT | -1 |
| FirstIn | Bit-coded number of the first signal that has resulted in a change at the output | DWORD | 16#00000000 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| Out | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 1976) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1976) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1976) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 1985) | DWORD | 16#00000000 |
| Status5 | Status word 5 (Page 1976) | DWORD | 16#00000000 |
| Status6 | Status word 6 (Page 1976) | DWORD | 16#00000000 |
| Status7 | Status word 7 (Page 1976) | DWORD | 16#00000000 |
| Status8 | Status word 8 (Page 1976) | DWORD | 16#00000000 |

**See also**

Intlk08 block diagram (Page 1991)
Intlk08 modes (Page 1979)

## 9.3.7 Intlk08 block diagram

### Intlk08 block diagram

A block diagram is not provided for this block.

**See also**

Intlk08 I/Os (Page 1985)
Intlk08 messaging (Page 1985)
Intlk08 error handling (Page 1984)
Intlk08 functions (Page 1980)
Intlk08 modes (Page 1979)
Description of Intlk08 (Page 1976)

## 9.3.8 Operator control and monitoring

### 9.3.8.1 Interlock block views

### Views of the blocks Intlk02, Intlk04, Intlk08, Intlk16

The blocks provide the following views:

- Interlock blocks standard view (Page 310)
- Preview of interlock blocks (Page 333)
- Batch view (Page 336)
- Block icon for interlock blocks (Page 272)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

# 9.4 Intlk16 - Interlock display with 16 input signals

## 9.4.1 Description of Intlk16

### Object name (type + number) and family

Type + number: FB 1827

Family: Interlck

### Area of application for Intlk16

The block is used for the following applications:

- Standardized interlock with display

### How it works

The block is used to calculate a standardized interlock that can be displayed on the OS. A maximum of 16 input signals can be supplied to the block. They are linked using selectable binary logic. The signal status of the output signal is also determined. You can assign an analog value with the signal status and unit to each input value for display in the faceplate.

The current state is displayed at the `Out` output parameter:

- `Out` = 0: Interlock
- `Out` = 1: "Good" state

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Intlk02 (Intlk16) block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Refer to Description of Intlk02 (Page 1949) for more information.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section Intlk16 I/Os (Page 2003).

| Status bit | Parameter |
|---|---|
| 0 | 1 = `Logic` = OR |
| 1 | 1 = `Logic` = AND |
| 2 | Not used |

| Status bit | Parameter |
|---|---|
| 3 | Result of logic operation `Out.Value` |
| 4 | 1 = At least one input value is excluded. See the section Excluding input values in Intlk16 functions (Page 1997) |
| 5 | 1 = All input values are excluded |
| 6 | 1 = No input value is interconnected or `NotUsed.Value` |
| 7 | Status display for simulation |
| 8 | Status display for not interlocked |
| 9 | Status display for interlocked |
| 10 | Not used |
| 11 | `Feature.Bit2`: Separate evaluation for excluded and simulated interlock signals |
| 12 | Internal error |
| 13-29 | Not used |
| 30 | `Occupied` |
| 31 | `BatchEn` |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `In01.Value` |
| 1 | `In02.Value` |
| 2 | `In03.Value` |
| 3 | `In04.Value` |
| 4 | `In05.Value` |
| 5 | `In06.Value` |
| 6 | `In07.Value` |
| 7 | `In08.Value` |
| 8 | `In09.Value` |
| 9 | `In10.Value` |
| 10 | `In11.Value` |
| 11 | `In12.Value` |
| 12 | `In13.Value` |
| 13 | `In14.Value` |
| 14 | `In15.Value` |
| 15 | `In16.Value` |
| 16 - 31 | Not used |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 | `InvIn01` |
| 1 | `InvIn02` |
| 2 | `InvIn03` |

| Status bit | Parameter |
|---|---|
| 3 | InvIn04 |
| 4 | InvIn05 |
| 5 | InvIn06 |
| 6 | InvIn07 |
| 7 | InvIn08 |
| 8 | InvIn09 |
| 9 | InvIn10 |
| 10 | InvIn11 |
| 11 | InvIn12 |
| 12 | InvIn13 |
| 13 | InvIn14 |
| 14 | InvIn15 |
| 15 | InvIn16 |
| 16 - 31 | Not used |

### Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0 | In01 with inversion |
| 1 | In02 with inversion |
| 2 | In03 with inversion |
| 3 | In04 with inversion |
| 4 | In05 with inversion |
| 5 | In06 with inversion |
| 6 | In07 with inversion |
| 7 | In08 with inversion |
| 8 | In09 with inversion |
| 9 | In10 with inversion |
| 10 | In11 with inversion |
| 11 | In12 with inversion |
| 12 | In13 with inversion |
| 13 | In14 with inversion |
| 14 | In15 with inversion |
| 15 | In16 with inversion |
| 16 | Bypass In01 (via interconnection) |
| 17 | Bypass In02 (via interconnection) |
| 18 | Bypass In03 (via interconnection) |
| 19 | Bypass In04 (via interconnection) |
| 20 | Bypass In05 (via interconnection) |
| 21 | Bypass In06 (via interconnection) |
| 22 | Bypass In07 (via interconnection) |
| 23 | Bypass In08 (via interconnection) |

| Status bit | Parameter |
|---|---|
| 24 | Bypass `In09` (via interconnection) |
| 25 | Bypass `In10` (via interconnection) |
| 26 | Bypass `In11` (via interconnection) |
| 27 | Bypass `In12` (via interconnection) |
| 28 | Bypass `In13` (via interconnection) |
| 29 | Bypass `In14` (via interconnection) |
| 30 | Bypass `In15` (via interconnection) |
| 31 | Bypass `In16` (via interconnection) |

### Status word allocation for `Status5` parameter

| Status bit | Parameter |
|---|---|
| 0 | `BypIn01` |
| 1 | `BypIn02` |
| 2 | `BypIn03` |
| 3 | `BypIn04` |
| 4 | `BypIn05` |
| 5 | `BypIn06` |
| 6 | `BypIn07` |
| 7 | `BypIn08` |
| 8 | `BypIn09` |
| 9 | `BypIn10` |
| 10 | `BypIn11` |
| 11 | `BypIn12` |
| 12 | `BypIn13` |
| 13 | `BypIn14` |
| 14 | `BypIn15` |
| 15 | `BypIn16` |
| 16 - 31 | `In01`..`In16`hidden bypass information |

### Status word allocation for `Status6` parameter

| Status bit | Parameter |
|---|---|
| 0 | `In01` not connected |
| 1 | `In02` not connected |
| 2 | `In03` not connected |
| 3 | `In04` not connected |
| 4 | `In05` not connected |
| 5 | `In06` not connected |
| 6 | `In07` not connected |
| 7 | `In08` not connected |
| 8 | `In09` not connected |

| Status bit | Parameter |
|---|---|
| 9 | `In10` not connected |
| 10 | `In11` not connected |
| 11 | `In12` not connected |
| 12 | `In13` not connected |
| 13 | `In14` not connected |
| 14 | `In15` not connected |
| 15 | `In16` not connected |
| 16 - 31 | Not used |

### Status word allocation for `Status7` parameter

| Status bit | Parameter |
|---|---|
| 0 | `AV01` not connected |
| 1 | `AV02` not connected |
| 2 | `AV03` not connected |
| 3 | `Av04` not connected |
| 4 | `AV05` not connected |
| 5 | `AV06` not connected |
| 6 | `AV07` not connected |
| 7 | `AV08` not connected |
| 8 | `AV09` not connected |
| 9 | `AV10` not connected |
| 10 | `AV11` not connected |
| 11 | `AV12` not connected |
| 12 | `AV13` not connected |
| 13 | `AV14` not connected |
| 14 | `AV15` not connected |
| 15 | `AV16` not connected |
| 16 - 31 | Not used |

### Status word allocation for `Status8` parameter

Identical to `FirstIn`.

### See also

Intlk16 block diagram (Page 2010)
Intlk16 error handling (Page 2002)
Intlk16 modes (Page 1997)
Intlk16 messaging (Page 2002)

## 9.4.2 Intlk16 modes

### Intlk16 modes

This block does not have any modes.

### See also

Intlk16 block diagram (Page 2010)
Intlk16 I/Os (Page 2003)
Intlk16 messaging (Page 2002)
Intlk16 functions (Page 1997)
Intlk16 error handling (Page 2002)
Description of Intlk16 (Page 1992)

## 9.4.3 Intlk16 functions

### Functions of Intlk16

The functions for this block are listed below.

### Logic operators

Use the `Logic` input to specify the logic operator that the block should employ when determining the interlock state. Make the following settings:

- `Logic` = 0: OR
- `Logic` = 1: AND

### Inversion of logic signals

You can invert the input signals by setting the input parameter `InvInx` for the input concerned to `Inx` = 1, e.g. for input `In01` set the I/O `InvIn01`.

The inversion is displayed in the faceplate. If you invert signals using any other method, this is not shown in the faceplate.

## Bypass

**Note**

Bypassing the interlock means that the interlock signal (input signal) is excluded from the logic of the interlock block, in other words, this signal is ignored in the logic operation.

This function can only be executed in the faceplate with "high-level operating permission".

You can bypass interlock signals as follows:

- From operator over faceplate (`BypInx` = 1):
  Independent `BypLix` or `Inx.Bit1` are connected and enabled, the operator can bypass or reset bypass the interlock input signal.
- By connecting `BypLix` in CFC (`BypLix.Value` = 1, `BypLix.ST` <> `16#FF`)
  Follow this if interlock signals are connected to outputs, which has no bypass information bit. Refer to sectionBypassing signals (Page 115).
- By bypass information bit in the connected interlock signal `Inx.Bit1`. For this the corresponding enable bit (`BypEn.Inx` = 1) and `Feature.Bit2` Separate evaluation for excluded and simulated interlock signals (Page 162) are to be set. Follow this if the interlock signals are connected to outputs which has the bypass information bit. Refer to sectionBypassing signals (Page 115).

The following table shows the possibilities to switch on/off the bypass of an interlock signal:

| BypLix.ST <> 16#FF | BypEn.Inx AND Feature.Bit2 | Interpretation |
|---|---|---|
| - | - | Only operator: `BypInx` |
| x | - | Input `BypLix` and operator |
| - | x | Bypass bit in Interlock input `Inx.Bit1` and operator |
| x | x | "`BypLix.Value OR Inx.Bit1`" and operator |

A value change of the used bypass signal `(BypLix.Value, Inx.Bit1)` tracks the input `BypInx` for faceplate and actualizes the bypass value. You can always operate the bypass over faceplate if no change of the used connected bypass inputs is active in block cycle.

**Note**

After a change of the parameterization in CFC (bypass via information bit `Bit1` in the interlock signals or bypass via interconnection separate inputs `BypLix`), a corresponding trigger of the bypass value must be made for the change to take effect.

You can also use both connected bypass inputs (`BypLix.Value, Inx.Bit1`) for one interlock input signal. Have a look that all prerequisites are fulfilled. Then with a change of `BypLix.Value` or `Inx.Bit1` the bypass of an interlock input `BypInx` is set to "`BypLix.Value OR Inx.Bit1`". Both inputs are to be false to reset the bypass. After that operator can change bypass over faceplate again.

The following symbol in standard view indicates a bypass request signal from the connected parameter.



- `Inx.Bit1 =1`: Output `Out.Bit1` of an interlock block,
  Output from a technological block with bypass functionality
- `BypLix.Value =1`: Output from a connected block

The following symbol in standard view indicates whether the interlock input signal is in bypass:



At the outputs `Byp01Act`... `Byp02Act`, you can see whether the corresponding input interlock signal is bypassed (`BypxAct` =`BypInx`).

You can reset all the bypass of interlock signals as follows:

- From operator over faceplate (`RstBypOp` = 0 -> 1)
- Over CFC linkable parameter (`RstBypLi` = 0 -> 1)

**Special situation:** If all input parameters are bypassed, the output value is defined using the `DefaultOut` parameter.

The output `BypAct` displays the information if there is an operated or connected bypass available.

```
BypAct.Value = BypIn01 OR BypInx …
                       OR In01.Bit1 OR Inx.Bit1 …
                       OR BypLi1.Value OR BypLix.Value …
```

**Note**

Do not connect the input `BypLix` with the output `BypAct` from an interlock block as `BypAct` is calculated from the internal bypass signal information and is only for display bypass and not for operate a bypass of an interlock input.

The bypass information `Out.Bit1` and the status `Out.ST` depends on `Feature.Bit2`:

**`Feature.Bit2` = 0:**

`Out.Bit1` = 0, Bypass information bit is not active and reset.
A bypassed interlock input, which is switch-relevant, sets the status of `Out.ST` to Simulation.

**`Feature.Bit2` = 1:**

`Out.Bit1 = BypAct.Value`, Bypass information bit is active and tracked.
A bypassed interlock input has no influence on the status of `Out.ST`

## Handling non-connected inputs

Inputs which are not interconnected are not evaluated. They are not displayed in the faceplate either.

**Special situation:** If no inputs are connected, the output value is defined using the `DefaultOut` parameter. The signal status is set to 16#FF.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234). However, only one additional faceplate can be called up using `SelFp1` = 1 .

### First-in detection for interlock blocks

This block provides the standard function Recording the first signal for interlock blocks (Page 58). Note that a signal status change only has an effect on the first-in detection when the `Feature` bit Evaluation of signal status (Page 152) and the input FirstInEn is set.

The response to deactivation via the `FirstInEn` input can be influenced by `Feature` bit First-in detection response to deactivation (Page 187).

**Note**

This function can only be executed in the faceplate with "process control" operating permission.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for interlock blocks (Page 124).

The worst signal status `ST_Worst` for the block is formed from the following parameter:

- `OUT.ST`

### Operating permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | 1 = Operator can exclude values |
| 1 | 1 = Operator can reset the exclusion of input values |
| 2 | 1 = Operator can reset the recording of the first signal |
| 3 - 15 | Not used |
| 16 | 1 = Operator can set or reset the exclusion of input value In01 |
| 17 | 1 = Operator can set or reset the exclusion of input value In02 |
| 18 | 1 = Operator can set or reset the exclusion of input value In03 |
| 19 | 1 = Operator can set or reset the exclusion of input value In04 |
| 20 | 1 = Operator can set or reset the exclusion of input value In05 |
| 21 | 1 = Operator can set or reset the exclusion of input value In06 |
| 22 | 1 = Operator can set or reset the exclusion of input value In07 |
| 23 | 1 = Operator can set or reset the exclusion of input value In08 |
| 24 | 1 = Operator can set or reset the exclusion of input value In09 |
| 25 | 1 = Operator can set or reset the exclusion of input value In10 |
| 26 | 1 = Operator can set or reset the exclusion of input value In11 |

| Bit | Function |
|---|---|
| 27 | 1 = Operator can set or reset the exclusion of input value In12 |
| 28 | 1 = Operator can set or reset the exclusion of input value In13 |
| 29 | 1 = Operator can set or reset the exclusion of input value In14 |
| 30 | 1 = Operator can set or reset the exclusion of input value In15 |
| 31 | 1 = Operator can set or reset the exclusion of input value In16 |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

## Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the chapter Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 2 | Separate evaluation for excluded and simulated interlock signals (Page 162) |
| 5 | Activate OS_Perm bits (Page 168)<br>0 = `OS_Perm` bits 16...17 inactive (evaluation only in faceplate)<br>1 = `OS_Perm` bits 16...17 active (evaluation only in faceplate) |
| 21 | First-in detection response to deactivation (Page 187) |
| 23 | Evaluation of signal status (Page 152) |
| 24 | Enabling local operator authorization (Page 169) |
| 31 | Activating recording of the first signal (Page 161) |

## SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

## See also

Intlk16 block diagram (Page 2010)

Intlk16 error handling (Page 2002)

Intlk16 modes (Page 1997)

Description of Intlk16 (Page 1992)

Intlk16 I/Os (Page 2003)

Intlk16 messaging (Page 2002)

## 9.4.4 Intlk16 error handling

### Error handling of Intlk16

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 13 | The parameter `Logic` has not been set to 0 or 1. |

### See also

Intlk16 block diagram (Page 2010)

Intlk16 I/Os (Page 2003)

Intlk16 messaging (Page 2002)

Intlk16 functions (Page 1997)

Intlk16 modes (Page 1997)

Description of Intlk16 (Page 1992)

## 9.4.5 Intlk16 messaging

### Messaging

This block does not offer messaging.

### See also

Intlk16 block diagram (Page 2010)

Intlk16 error handling (Page 2002)

Intlk16 modes (Page 1997)

Description of Intlk16 (Page 1992)

Intlk16 I/Os (Page 2003)
Intlk16 functions (Page 1997)

## 9.4.6 Intlk16 I/Os

I/Os of Intlk16

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AV01 | Analog value of In01 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV02 | Analog value of In02 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV03 | Analog value of In03 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV04 | Analog value of In04 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV05 | Analog value of In05 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV06 | Analog value of In06 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV07 | Analog value of In07 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV08 | Analog value of In08 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV09 | Analog value of In09 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV10 | Analog value of In10 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| AV11 | Analog value of In11 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV12 | Analog value of In12 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV13 | Analog value of In13 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV14 | Analog value of In14 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV15 | Analog value of In15 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV16 | Analog value of In16 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV01_Unit | Unit of measure for AV01 | INT | 0 |
| AV02_Unit | Unit of measure for AV02 | INT | 0 |
| AV03_Unit | Unit of measure for AV03 | INT | 0 |
| AV04_Unit | Unit of measure for AV04 | INT | 0 |
| AV05_Unit | Unit of measure for AV05 | INT | 0 |
| AV06_Unit | Unit of measure for AV06 | INT | 0 |
| AV07_Unit | Unit of measure for AV07 | INT | 0 |
| AV08_Unit | Unit of measure for AV08 | INT | 0 |
| AV09_Unit | Unit of measure for AV09 | INT | 0 |
| AV10_Unit | Unit of measure for AV10 | INT | 0 |
| AV11_Unit | Unit of measure for AV11 | INT | 0 |
| AV12_Unit | Unit of measure for AV12 | INT | 0 |
| AV13_Unit | Unit of measure for AV13 | INT | 0 |
| AV14_Unit | Unit of measure for AV14 | INT | 0 |
| AV15_Unit | Unit of measure for AV15 | INT | 0 |
| AV16_Unit | Unit of measure for AV16 | INT | 0 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BypIn01* | 1 = Input In01 is not used | BOOL | 0 |
| BypIn02* | 1 = Input In02 is not used | BOOL | 0 |
| BypIn03* | 1 = Input In03 is not used | BOOL | 0 |
| BypIn04* | 1 = Input In04 is not used | BOOL | 0 |
| BypIn05* | 1 = Input In05 is not used | BOOL | 0 |
| BypIn06* | 1 = Input In06 is not used | BOOL | 0 |
| BypIn07* | 1 = Input In07 is not used | BOOL | 0 |
| BypIn08* | 1 = Input In08 is not used | BOOL | 0 |
| BypIn09* | 1 = Input In09 is not used | BOOL | 0 |
| BypIn10* | 1 = Input In10 is not used | BOOL | 0 |
| BypIn11* | 1 = Input In11 is not used | BOOL | 0 |
| BypIn12* | 1 = Input In12 is not used | BOOL | 0 |
| BypIn13* | 1 = Input In13 is not used | BOOL | 0 |
| BypIn14* | 1 = Input In14 is not used | BOOL | 0 |
| BypIn15* | 1 = Input In15 is not used | BOOL | 0 |
| BypIn16* | 1 = Input In16 is not used | BOOL | 0 |
| BypLi01...BypLi16 | Bypass In01...In16 (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| DefaultOut | Output value for cases where all inputs are excluded or not interconnected. Refer to the section Intlk16 functions (Page 1997) of the block. | BOOL | 1 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 1997) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| FirstInEn | First-in detection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| `In01` | Input `In01` | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| `In02` | Input `In02` | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| `In03` | Input `In03` | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| `In04` | Input `In04` | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| `In05` | Input `In05` | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| `In06` | Input `In06` | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| `In07` | Input `In07` | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| In08 | Input In08 | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| In09 | Input In09 | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| In10 | Input In10 | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| In11 | Input In11 | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| In12 | Input In12 | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| In13 | Input In13 | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| In14 | Input In14 | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| In15 | Input In15 | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| In16 | Input In16 | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#FF<br>• 0 |
| InvIn01 | Invert input In01 | BOOL | 0 |
| InvIn02 | Invert input In02 | BOOL | 0 |
| InvIn03 | Invert input In03 | BOOL | 0 |
| InvIn04 | Invert input In04 | BOOL | 0 |
| InvIn05 | Invert input In05 | BOOL | 0 |
| InvIn06 | Invert input In06 | BOOL | 0 |
| InvIn07 | Invert input In07 | BOOL | 0 |
| InvIn08 | Invert input In08 | BOOL | 0 |
| InvIn09 | Invert input In09 | BOOL | 0 |
| InvIn10 | Invert input In10 | BOOL | 0 |
| InvIn11 | Invert input In11 | BOOL | 0 |
| InvIn12 | Invert input In12 | BOOL | 0 |
| InvIn13 | Invert input In13 | BOOL | 0 |
| InvIn14 | Invert input In14 | BOOL | 0 |
| InvIn15 | Invert input In15 | BOOL | 0 |
| InvIn16 | Invert input In16 | BOOL | 0 |
| Logic | Logical operation:<br>0 = Logical OR<br>1 = Logical AND | INT | 0 |
| NotUsed | 1 = Block is not used (only for display in the faceplate). You can find additional information on this in the section Interlock blocks standard view (Page 310) / Color of the field. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OS_Perm | I/O for operating permissions (Page 1997) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| RstBypOp* | 1 = Reset inputs BypIn01 to BypIn16 | BOOL | 0 |
| RstLi | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | 1 = Reset via operator | BOOL | 0 |
| SelFp1 | Call a block saved in this parameter as Opening additional faceplates (Page 234) in standard view | ANY | |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| BypAct | 1 = Bypass active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Byp01Act | 1 = Bypass of input 01 is active | BOOL | 0 |
| Byp02Act | 1 = Bypass of input 02 is active | BOOL | 0 |
| Byp03Act | 1 = Bypass of input 03 is active | BOOL | 0 |
| Byp04Act | 1 = Bypass of input 04 is active | BOOL | 0 |
| Byp05Act | 1 = Bypass of input 05 is active | BOOL | 0 |
| Byp06Act | 1 = Bypass of input 06 is active | BOOL | 0 |
| Byp07Act | 1 = Bypass of input 07 is active | BOOL | 0 |
| Byp08Act | 1 = Bypass of input 08 is active | BOOL | 0 |
| Byp09Act | 1 = Bypass of input 09 is active | BOOL | 0 |
| Byp10Act | 1 = Bypass of input 10 is active | BOOL | 0 |
| Byp11Act | 1 = Bypass of input 11 is active | BOOL | 0 |
| Byp12Act | 1 = Bypass of input 12 is active | BOOL | 0 |
| Byp13Act | 1 = Bypass of input 13 is active | BOOL | 0 |
| Byp14Act | 1 = Bypass of input 14 is active | BOOL | 0 |
| Byp15Act | 1 = Bypass of input 15 is active | BOOL | 0 |
| Byp16Act | 1 = Bypass of input 16 is active | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Out | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 1992) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 1992) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 1992) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 1992) | DWORD | 16#00000000 |
| Status5 | Status word 5 (Page 1992) | DWORD | 16#00000000 |
| Status6 | Status word 6 (Page 1992) | DWORD | 16#00000000 |
| Status7 | Status word 7 (Page 1992) | DWORD | 16#00000000 |
| Status8 | Status word 8 (Page 1992) | DWORD | 16#00000000 |

### See also

Intlk16 block diagram (Page 2010)
Intlk16 error handling (Page 2002)
Intlk16 modes (Page 1997)
Intlk16 messaging (Page 2002)

## 9.4.7 Intlk16 block diagram

### Intlk16 block diagram

A block diagram is not provided for this block.

### See also

Intlk16 I/Os (Page 2003)
Intlk16 messaging (Page 2002)
Intlk16 functions (Page 1997)
Intlk16 error handling (Page 2002)
Intlk16 modes (Page 1997)
Description of Intlk16 (Page 1992)

## 9.4.8 Operator control and monitoring

### 9.4.8.1 Interlock block views

#### Views of the blocks Intlk02, Intlk04, Intlk08, Intlk16

The blocks provide the following views:

- Interlock blocks standard view (Page 310)
- Preview of interlock blocks (Page 333)
- Batch view (Page 336)
- Block icon for interlock blocks (Page 272)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

# 9.5 FirstIn - Transform output Intlck FirstIn for message associated value

## 9.5.1 Description of FirstIn

### Object name (type + number) and family

Type + number: FC 1929

Family: Interlck

### Area of application for FirstIn

The block is used for the following applications:

- To generate an external message that can be seen in the technological block.
- To generate an external message from EventTs or Event16Ts block.

### How it works

FirstIn block works only with the combination of a technological block that has `ExtMsgx` and `ExtValxx` input parameters.

The block generates messages requiring acknowledgment that appear in the alarm view of the interconnected technological block.

The block must be connected to a technological block. Messages of the interconnected technological blocks are displayed in the message view.

FirstIn block also works with EventTs and Event16Ts blocks to generate an external message. External signals of FirstIn block should be connected with EventTs or Event16Ts block. When a fault occurs, EventTs or Event16Ts block is triggered and the message can be viewed in the interconnected technological block.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Connect the following output parameters with the input parameters as indicated in the following table:

| Output parameter | | Input parameter | |
|---|---|---|---|
| **Parameter name** | **Block** | **Parameter name** | **Block** |
| `FirstIn` | Interlock | `FirstIn` | FirstIn |
| `Out` | Interlock | `Interlock` | Technological |
| `FirstInAct` | FirstIn | `ExtMsgx` | Technological |
| `ExtVa1FI` | FirstIn | `ExtValxx` | Technological |
| `ExtVa2FI` | FirstIn | `ExtValxx` | Technological |

For example:

- Output parameters `ExtVa1FI` and `ExtVa2FI` are connected to the input parameters `ExtVa104` and `ExtVa108` respectively.
- Output parameter `FirstInAct` is connected to the input parameter `ExtMsg2`.

In the PCS 7 special message property of the external message 2 signal, add the following syntax:

"@4%s@@8%s@".

**With EventTs/Event16Ts blocks:**

You can connect FirstIn block with the block EventTs or Event16Ts. Connect the input parameters `ExtVaxxx` (`ExtVal01`...`ExtVal03` in case of EventTs block and `ExtVal01`...`ExtVa106` in case of Event16Ts block) with the output parameters `ExtVa1FI` or `ExtVa2FI` of the FirstIn block.

The output parameter `FirstInAct` of the FirstIn block should be connected with any `Inx` input parameters of the EventTs/Event16Ts blocks.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

FirstIn modes (Page 2013)
FirstIn functions (Page 2014)
FirstIn error handling (Page 2014)
FirstIn messaging (Page 2015)
FirstIn I/Os (Page 2015)
FirstIn block diagram (Page 2016)

## 9.5.2 FirstIn modes

### FirstIn operating modes

This block does not have any operating modes.

**See also**

Description of FirstIn (Page 2012)
FirstIn functions (Page 2014)
FirstIn error handling (Page 2014)
FirstIn messaging (Page 2015)
FirstIn I/Os (Page 2015)
FirstIn block diagram (Page 2016)

## 9.5.3 FirstIn functions

**Functions of FirstIn**

This block does not have any special function.

**See also**

Description of FirstIn (Page 2012)
FirstIn modes (Page 2013)
FirstIn error handling (Page 2014)
FirstIn messaging (Page 2015)
FirstIn I/Os (Page 2015)
FirstIn block diagram (Page 2016)

## 9.5.4 FirstIn error handling

**Error handling of FirstIn**

This block does not have any error handling.

**See also**

Description of FirstIn (Page 2012)
FirstIn modes (Page 2013)
FirstIn functions (Page 2014)
FirstIn messaging (Page 2015)
FirstIn I/Os (Page 2015)
FirstIn block diagram (Page 2016)

## 9.5.5 FirstIn messaging

### Messaging

This block does not offer messaging.

### See also

Description of FirstIn (Page 2012)
FirstIn modes (Page 2013)
FirstIn functions (Page 2014)
FirstIn error handling (Page 2014)
FirstIn I/Os (Page 2015)
FirstIn block diagram (Page 2016)

## 9.5.6 FirstIn I/Os

### I/Os of FirstIn

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| FirstIn | First active input | DWORD | 16#00000000 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| FirstInAct | First signal active for ExtMsgX or EventTs message | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ExtVa1FI | | STRUCT<br>• Char1: CHAR<br>• ...<br>• Char15: CHAR | -<br>• ''<br>• ''<br>• '' |
| ExtVa2FI | | STRUCT<br>• Char1: CHAR<br>• ...<br>• Char23: CHAR | -<br>• ''<br>• ''<br>• '' |

### See also

Description of FirstIn (Page 2012)
FirstIn modes (Page 2013)
FirstIn functions (Page 2014)
FirstIn error handling (Page 2014)
FirstIn messaging (Page 2015)
FirstIn block diagram (Page 2016)

## 9.5.7 FirstIn block diagram

### Block diagram of FirstIn

A block diagram is not provided for this block.

### See also

Description of FirstIn (Page 2012)
FirstIn modes (Page 2013)
FirstIn functions (Page 2014)
FirstIn error handling (Page 2014)
FirstIn messaging (Page 2015)
FirstIn I/Os (Page 2015)

# Message blocks 10

## 10.1 Event - Creating messages

### 10.1.1 Description of Event

#### Object name (type + number) and family

Type + number: FB 1811

Family: Report

#### Area of application for Event

The block is used for the following applications:

- Generation of messages requiring acknowledgment

#### How it works

The block is used to simultaneously output up to eight different messages that require acknowledgment.

The individual messages are assigned to the monitored signals via the inputs and the messages released or blocked depending on the process status data. If a change is made to one or more of the monitored signals enabled for reporting, a message is output.

The signals to be monitored are interconnected with inputs `In1` ... `In8`. Each `Inx` signal can also be inverted via input `InvInx`. A message is output if a signal value is changed (but taking inversion into account).

Each input is assigned a separate message text. A message received at input `In5`, for example, is output with the message text for the SIG 5 signal.

#### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

Further addressing is not required.

#### Startup characteristics

During startup, the message block suppresses all messages. The duration (number of cycles) of message suppression is set at the `RunUpCyc` parameter. During restart (OB100) an internal counter that is initialized with this value decrements each time the block is executed. Messages are not generated if the counter value does not equal zero.

Messages whose alarm delay has not elapsed during this period are then output.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section Event I/Os (Page 2025).

| Status bit | Parameter |
|---|---|
| 0 | `In1.Value` |
| 1 | `In2.Value` |
| 2 | `In3.Value` |
| 3 | `In4.Value` |
| 4 | `In5.Value` |
| 5 | `In6.Value` |
| 6 | `In7.Value` |
| 7 | `In8.Value` |
| 8 | `InvIn1` |
| 9 | `InvIn2` |
| 10 | `InvIn3` |
| 11 | `InvIn4` |
| 12 | `InvIn5` |
| 13 | `InvIn6` |
| 14 | `InvIn7` |
| 15 | `InvIn8` |
| 16 | `In1` with inversion |
| 17 | `In2` with inversion |
| 18 | `In3` with inversion |
| 19 | `In4` with inversion |
| 20 | `In5` with inversion |
| 21 | `In6` with inversion |
| 22 | `In7` with inversion |
| 23 | `In8` with inversion |
| 24 | `In1` not connected |
| 25 | `In2` not connected |
| 26 | `In3` not connected |
| 27 | `In4` not connected |
| 28 | `In5` not connected |
| 29 | `In6` not connected |
| 30 | `In7` not connected |
| 31 | `In8` not connected |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `In1MsgEn` |
| 1 | `In2MsgEn` |
| 2 | `In3MsgEn` |
| 3 | `In4MsgEn` |
| 4 | `In5MsgEn` |
| 5 | `In6MsgEn` |
| 6 | `In7MsgEn` |
| 7 | `In8MsgEn` |
| 8 | `AV1` not connected |
| 9 | `AV2` not connected |
| 10 | `AV3` not connected |
| 11 | `AV4` not connected |
| 12 | `AV5` not connected |
| 13 | `AV6` not connected |
| 14 | `AV7` not connected |
| 15 | `AV8` not connected |
| 16 | Active signal 1 for messages |
| 17 | Active signal 2 for messages |
| 18 | Active signal 3 for messages |
| 19 | Active signal 4 for messages |
| 20 | Active signal 5 for messages |
| 21 | Active signal 6 for messages |
| 22 | Active signal 7 for messages |
| 23 | Active signal 8 for messages |
| 24 | `MsgLock` |
| 25 | `Occupied` |
| 26 | `BatchEn` |
| 27 | Batch - parameter exists |
| 28 - 31 | Not used |

### See also

Event functions (Page 2020)
Event messaging (Page 2023)
Event block diagram (Page 2029)
Event error handling (Page 2022)
Event modes (Page 2020)

## 10.1.2 Event modes

### Event operating modes

The block can be operated using the following modes:

- On (Page 77)
- Out of service (Page 77)

### "On"

You can find general information about the "On" mode in the On (Page 77) section.

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

### See also

Event block diagram (Page 2029)

Event I/Os (Page 2025)

Event messaging (Page 2023)

Event error handling (Page 2022)

Event functions (Page 2020)

Description of Event (Page 2017)

## 10.1.3 Event functions

### Functions of Event

The functions for this block are listed below.

### Activation and deactivation of messages

Set the I/Os `In1MsgEn` to `In8MsgEn` accordingly to activate or deactivate the messages applied to inputs `In1` to `In8`. All messages are activated by default.

To deactivate messages received at I/O `In4`, for example, you set I/O `In4MsgEn` = 0 accordingly.

You can deactivate all messages via I/O `MsgLock` = 1 .

### Delay of alarms

You can delay the alarms indicating signal changes.

For incoming alarms (signal change 0 - 1), set the delay at parameter `AlmOnDly`; for outgoing alarms (signal change 1 - 0) set it at parameter `AlmOffDly`.

Enter 0 or a negative value to deactivate the delay.

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Operator control permissions

The block has the following Operator control permissions (Page 285) for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 - 31 | Not used |

This block does not have a faceplate yet; the operator control permissions have already been assigned in the planning phase for these faceplates.

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the section Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 1 | Reaction to the out of service mode (Page 188) |
| 8 | Reporting with BATCH parameters (Page 167) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 27 | Selecting values associated with messages (Page 167) |

### Forming the signal status for blocks

The block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The block determines the worst signal status via all interconnected binary and analog inputs, and outputs this value at `ST_Worst`.

- `In1`

etc. to

- `In8`
- `AV1`

etc. to

- `AV8`

### See also

Description of Event (Page 2017)

Event messaging (Page 2023)

Event I/Os (Page 2025)

Event block diagram (Page 2029)

Event error handling (Page 2022)

Event modes (Page 2020)

## 10.1.4 Event error handling

### Error handling of Event

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |

### See also

Event block diagram (Page 2029)

Event I/Os (Page 2025)

Event messaging (Page 2023)

Event functions (Page 2020)

Event modes (Page 2020)

Description of Event (Page 2017)

## 10.1.5 Event messaging

### Messaging

Messages which can be acknowledged are generated using ALARM_8P. The block uses the PMC communication channel and has 8 digital inputs and 8 associated values.

Every edge transition detected for one or more digital inputs results in a message. The associated values are assigned consistently to the message at the time of edge evaluation. All 8 signals are assigned a common message number, which is split at the OS into 8 messages. The Engineering System (ES) assigns the message number automatically by calling the message server.

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId` | SIG 1 | AS process control message - error | Text 1 |
| | SIG 2 | AS process control message - error | Text 2 |
| | SIG 3 | AS process control message - error | Text 3 |
| | SIG 4 | AS process control message - error | Text 4 |
| | SIG 5 | AS process control message - error | Text 5 |
| | SIG 6 | AS process control message - error | Text 6 |
| | SIG 7 | AS process control message - error | Text 7 |
| | SIG 8 | AS process control message - error | Text 8 |

You can change the message class and the event to meet your needs at the block type and/or block instance.

Depending on `Feature.Bit27` "Select message associated values", either the signal status or the associated analog value is written as message associated value ( `Feature.Bit8` = 0).

### Associated values for message instance `MsgEvId` (`Feature.Bit27` = 0)

| Associated value | Block parameters |
|---|---|
| 1 | `In1.ST` |
| 2 | `In2.ST` |
| 3 | `In3.ST` |
| 4 | `In4.ST` |
| 5 | `In5.ST` |
| 6 | `In6.ST` |
| 7 | `In7.ST` |
| 8 | `In8.ST` |
| 9 | Not allocated |
| 10 | Not allocated |

### Associated values for message instance `MsgEvId` (`Feature.Bit27` = 1)

| Associated value | Block parameters |
|---|---|
| 1 | `AV1.Value` |
| 2 | `AV2.Value` |
| 3 | `AV3.Value` |
| 4 | `AV4.Value` |
| 5 | `AV5.Value` |
| 6 | `AV6.Value` |
| 7 | `AV7.Value` |
| 8 | `AV8.Value` |
| 9 | Not allocated |
| 10 | Not allocated |

### The batch information is transmitted with `Feature.Bit8` = 1:

The first three associated values are described as follows and are followed by the signal status of the input signal or the associated analog value, depending on `Feature.Bit27` "Selecting message associated values":

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchId |
| 4 | In1.ST / AV1.Value |
| 5 | In2.ST / AV2.Value |
| 6 | In3.ST / AV3.Value |
| 7 | In4.ST / AV4.Value |
| 8 | In5.ST / AV5.Value |

| Associated value | Block parameters |
|---|---|
| 9 | In6.ST / AV6.Value |
| 10 | In7.ST / AV7.Value |

Enter the batch ID @1%s@ under "Properties - Block - Special properties - Messages extended - Message texts block".

### See also

Description of Event (Page 2017)

Event functions (Page 2020)

Event I/Os (Page 2025)

Event block diagram (Page 2029)

Event error handling (Page 2022)

Event modes (Page 2020)

Selecting values associated with messages (Page 167)

## 10.1.6 Event I/Os

### I/Os of Event

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AlmOnDly | Alarm delay time [s] for signal transition 0 → 1 | REAL | 0.0 |
| AlmOffDly | Alarm delay time [s] for signal transition 1 → 0 | REAL | 0.0 |
| AV1 | Message associated value for In1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV1_Unit | Unit for AV1 | INT | 0 |
| AV2 | Message associated value for In2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV2_Unit | Unit for AV2 | INT | 0 |
| AV3 | Message associated value for In3 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| AV3_Unit | Unit for AV3 | INT | 0 |
| AV4 | Message associated value for In4 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| AV4_Unit | Unit for AV4 | INT | 0 |
| AV5 | Message associated value for In5 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| AV5_Unit | Unit for AV5 | INT | 0 |
| AV6 | Message associated value for In6 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| AV6_Unit | Unit for AV6 | INT | 0 |
| AV7 | Message associated value for In7 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| AV7_Unit | Unit for AV7 | INT | 0 |
| AV8 | Message associated value for In8 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| AV8_Unit | Unit for AV8 | INT | 0 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00 |
| BatchName | Batch name | S7-String | |
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 2020) | STRUCT | - |
| | | • Bit 0: BOOL | • 0 |
| | | • ... | • 0 |
| | | • Bit 31: BOOL | • 0 |
| In1 | Input In1 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#FF |
| In1MsgEn | 1 = Activate message for input In1 | BOOL | 1 |
| In2 | Input In2 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#FF |
| In2MsgEn | 1 = Activate message for input In2 | BOOL | 1 |
| In3 | Input In3 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#FF |
| In3MsgEn | 1 = Activate message for input In3 | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| In4 | Input In4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In4MsgEn | 1 = Activate message for input In4 | BOOL | 1 |
| In5 | Input In5 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In5MsgEn | 1 = Activate message for input In5 | BOOL | 1 |
| In6 | Input In6 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In6MsgEn | 1 = Activate message for input In6 | BOOL | 1 |
| In7 | Input In7 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In7MsgEn | 1 = Activate message for input In7 | BOOL | 1 |
| In8 | Input In8 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In8MsgEn | 1 = Activate message for input In8 | BOOL | 1 |
| InvIn1 | 1 = Invert input In1 | BOOL | 0 |
| InvIn2 | 1 = Invert input In2 | BOOL | 0 |
| InvIn3 | 1 = Invert input In3 | BOOL | 0 |
| InvIn4 | 1 = Invert input In4 | BOOL | 0 |
| InvIn5 | 1 = Invert input In5 | BOOL | 0 |
| InvIn6 | 1 = Invert input In6 | BOOL | 0 |
| InvIn7 | 1 = Invert input In7 | BOOL | 0 |
| InvIn8 | 1 = Invert input In8 | BOOL | 0 |
| MS_RelOp | 1 = Release for maintenance by OS operator | BOOL | 0 |
| MsgEvId | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>Refer also to the Suppressing messages using the MsgLock parameter (Page 231) section for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = In use by a batch | BOOL | 0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OS_Perm | I/O for operator control permissions (Page 2020) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUp-Cyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | RESERVED | ANY | |
| SelFp2 | RESERVED | ANY | |
| StepNo | Batch step number | DWORD | 16#00 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see Event error handling (Page 2022) | INT | -1 |
| MsgAckn | Message acknowledgment status (output ACK_STATE of ALARM_8P) | WORD | 16#0000 |
| MsgErr | 1 = Alarm error (output ERROR of ALARM_8) | BOOL | 0 |
| MsgStat | Alarm status (output STATUS of ALARM_8P) | WORD | 16#0000 |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| ST_Worst | Worst signal status | BYTE | 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Status1 | Status word 1 (Page 2017) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 2017) | DWORD | 16#00000000 |

**See also**

Event messaging (Page 2023)
Event block diagram (Page 2029)
Event modes (Page 2020)
Opening additional faceplates (Page 234)

## 10.1.7 Event block diagram

**Event block diagram**

A block diagram is not provided for this block.

**See also**

Event I/Os (Page 2025)
Event messaging (Page 2023)
Event error handling (Page 2022)
Event functions (Page 2020)
Event modes (Page 2020)
Description of Event (Page 2017)

# 10.2 EventNck - Generating messages without acknowledgment

## 10.2.1 Description of EventNck

### Object name (type + number) and family

Type + number: FB 1904

Family: Report

### Area of application for EventNck

The block is used for the following applications:

- Generation of messages not requiring acknowledgment

### How it works

The block is used to simultaneously output up to eight different messages that do not require acknowledgment.

The individual messages are assigned to the monitored signals via the inputs and the messages released or blocked depending on the process status data. If a change is made to one or more of the monitored signals enabled for reporting, a message is output.

The signals to be monitored are interconnected with inputs `In1` to `In8`. Each `Inx` signal can also be inverted via input `InvInx`. A message is output if a signal value is changed (but taking inversion into account).

Each input is assigned a separate message text. A message received at input `In5`, for example, is output with the message text for the SIG 5 signal.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

Further addressing is not required.

### Startup characteristics

During startup, the message block suppresses all messages. The duration (number of cycles) of message suppression is set at the `RunUpCyc` parameter. During restart (OB100) an internal counter that is initialized with this value decrements each time the block is executed. Messages are not generated if the counter value does not equal zero.

Messages whose alarm delay has not elapsed during this period are then output.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section EventNck I/Os (Page 2037).

| Status bit | Parameter |
|---|---|
| 0 | `In1.Value` |
| 1 | `In2.Value` |
| 2 | `In3.Value` |
| 3 | `In4.Value` |
| 4 | `In5.Value` |
| 5 | `In6.Value` |
| 6 | `In7.Value` |
| 7 | `In8.Value` |
| 8 | `InvIn1` |
| 9 | `InvIn2` |
| 10 | `InvIn3` |
| 11 | `InvIn4` |
| 12 | `InvIn5` |
| 13 | `InvIn6` |
| 14 | `InvIn7` |
| 15 | `InvIn8` |
| 16 | `In1` with inversion |
| 17 | `In2` with inversion |
| 18 | `In3` with inversion |
| 19 | `In4` with inversion |
| 20 | `In5` with inversion |
| 21 | `In6` with inversion |
| 22 | `In7` with inversion |
| 23 | `In8` with inversion |
| 24 | `In1` not connected |
| 25 | `In2` not connected |
| 26 | `In3` not connected |
| 27 | `In4` not connected |
| 28 | `In5` not connected |
| 29 | `In6` not connected |
| 30 | `In7` not connected |
| 31 | `In8` not connected |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `In1MsgEn` |
| 1 | `In2MsgEn` |
| 2 | `In3MsgEn` |

| Status bit | Parameter |
|---|---|
| 3 | `In4MsgEn` |
| 4 | `In5MsgEn` |
| 5 | `In6MsgEn` |
| 6 | `In7MsgEn` |
| 7 | `In8MsgEn` |
| 8 | `AV1` not connected |
| 9 | `AV2` not connected |
| 10 | `AV3` not connected |
| 11 | `AV4` not connected |
| 12 | `AV5` not connected |
| 13 | `AV6` not connected |
| 14 | `AV7` not connected |
| 15 | `AV8` not connected |
| 16 | Active signal 1 for messages |
| 17 | Active signal 2 for messages |
| 18 | Active signal 3 for messages |
| 19 | Active signal 4 for messages |
| 20 | Active signal 5 for messages |
| 21 | Active signal 6 for messages |
| 22 | Active signal 7 for messages |
| 23 | Active signal 8 for messages |
| 24 | `MsgLock` |
| 25 - 31 | Not used |

### See also

EventNck modes (Page 2032)
EventNck functions (Page 2033)
EventNck error handling (Page 2035)
EventNck messaging (Page 2036)
EventNck block diagram (Page 2041)

## 10.2.2 EventNck modes

### EventNck modes

The block provides the following modes:

- On (Page 77)
- Out of service (Page 77)

**"On"**

General information on the "On" mode is available in the section On (Page 77).

**"Out of service"**

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

**See also**

Description of EventNck (Page 2030)

EventNck functions (Page 2033)

EventNck error handling (Page 2035)

EventNck messaging (Page 2036)

EventNck I/Os (Page 2037)

EventNck block diagram (Page 2041)

## 10.2.3 EventNck functions

**Functions of EventNck**

The functions for this block are listed below.

**Activation and deactivation of messages**

Set the I/Os `In1MsgEn` ... `In8MsgEn` accordingly to enable or disable the messages applied to inputs `In1` ... `In8`. All messages are activated by default.

To deactivate messages received at I/O `In4`, for example, you set I/O `In4MsgEn` = 0 accordingly.

You can deactivate all messages via I/O `MsgLock` = 1.

**Delay of alarms**

You can delay the alarms indicating signal changes.

For incoming alarms (signal change 0 - 1), set the delay at parameter `AlmOnDly`; for outgoing alarms (signal change 1 - 0) set it at parameter `AlmOffDly`.

Enter 0 or a negative value to deactivate the delay.

**Suppressing messages using the `MsgLock` parameter**

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Operator control permissions

The block has the following Operator control permissions (Page 285) for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 - 31 | Not used |

This block does not have a faceplate yet; the operator control permissions have already been assigned in the planning phase for these faceplates.

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the section Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 1 | Reaction to the out of service mode (Page 188) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 27 | Selecting values associated with messages (Page 167) |

### Forming the signal status for blocks

The block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The block determines the worst signal status via all interconnected binary and analog inputs, and outputs this value at `ST_Worst`.

- `In1`

etc. to

- `In8`
- `AV1`

etc. to

- AV8

### See also

Description of EventNck (Page 2030)

EventNck modes (Page 2032)

EventNck error handling (Page 2035)

EventNck messaging (Page 2036)

EventNck I/Os (Page 2037)

EventNck block diagram (Page 2041)

## 10.2.4 EventNck error handling

### Error handling of EventNck

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The ErrorNum I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |

### See also

Description of EventNck (Page 2030)

EventNck modes (Page 2032)

EventNck functions (Page 2033)

EventNck messaging (Page 2036)

EventNck I/Os (Page 2037)

EventNck block diagram (Page 2041)

## 10.2.5 EventNck messaging

### Messaging

Messages not requiring acknowledgment are generated with NOTIFY_8P . The block uses the PMC communication channel and has 8 digital inputs and 10 associated values.

Every edge transition detected for one or more digital inputs results in a message. The associated values are assigned consistently to the message at the time of edge evaluation. All 8 signals are assigned a common message number, which is split at the OS into 8 messages. The Engineering System (ES) assigns the message number automatically by calling the message server.

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId` | SIG 1 | Operating message – without acknowledgment | Text 1 |
| | SIG 2 | Operating message – without acknowledgment | Text 2 |
| | SIG 3 | Operating message – without acknowledgment | Text 3 |
| | SIG 4 | Operating message – without acknowledgment | Text 4 |
| | SIG 5 | Operating message – without acknowledgment | Text 5 |
| | SIG 6 | Operating message – without acknowledgment | Text 6 |
| | SIG 7 | Operating message – without acknowledgment | Text 7 |
| | SIG 8 | Operating message – without acknowledgment | Text 8 |

You can change the message class and the event to meet your needs at the block type and/or block instance.

### Associated values for message instance `MsgEvId`

You can use the `Feature` Bit Selecting values associated with messages (Page 167) to define whether the signal status of the signal or the associated analog value is to be used as the associated value for the message.

| Associated value | Block parameters |
|---|---|
| 1 | `In1.ST` |
| 2 | `In2.ST` |
| 3 | `In3.ST` |
| 4 | `In4.ST` |

| Associated value | Block parameters |
|---|---|
| 5 | `In5.ST` |
| 6 | `In6.ST` |
| 7 | `In7.ST` |
| 8 | `In8.ST` |
| 9 | Not allocated |
| 10 | Not allocated |

### Associated values for message instance `MsgEvId`

| Associated value | Block parameters |
|---|---|
| 1 | `AV1.Value` |
| 2 | `AV2.Value` |
| 3 | `AV3.Value` |
| 4 | `AV4.Value` |
| 5 | `AV5.Value` |
| 6 | `AV6.Value` |
| 7 | `AV7.Value` |
| 8 | `AV8.Value` |
| 9 | Not allocated |
| 10 | Not allocated |

### See also

Description of EventNck (Page 2030)

EventNck modes (Page 2032)

EventNck functions (Page 2033)

EventNck error handling (Page 2035)

EventNck I/Os (Page 2037)

EventNck block diagram (Page 2041)

## 10.2.6 EventNck I/Os

### I/Os of EventNck

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

## Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AlmOnDly | Alarm delay time [s] for signal transition 0 → 1 | REAL | 0.0 |
| AlmOffDly | Alarm delay time [s] for signal transition 1 → 0 | REAL | 0.0 |
| AV1 | Message associated value for In1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV1_Unit | Unit for AV1 | INT | 0 |
| AV2 | Message associated value for In2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV2_Unit | Unit for AV2 | INT | 0 |
| AV3 | Message associated value for In3 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV3_Unit | Unit for AV3 | INT | 0 |
| AV4 | Message associated value for In4 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV4_Unit | Unit for AV4 | INT | 0 |
| AV5 | Message associated value for In5 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV5_Unit | Unit for AV5 | INT | 0 |
| AV6 | Message associated value for In6 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV6_Unit | Unit for AV6 | INT | 0 |
| AV7 | Message associated value for In7 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV7_Unit | Unit for AV7 | INT | 0 |
| AV8 | Message associated value for In8 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| AV8_Unit | Unit for AV8 | INT | 0 |
| EN | 1 = Called block will be processed | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Feature | I/O for additional functions (Page 2033) | STRUCT | - |
| | | • Bit 0: BOOL | • 0 |
| | | • ... | • 0 |
| | | • Bit 31: BOOL | • 0 |
| In1 | Input In1 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#FF |
| In1MsgEn | 1 = Activate message for input In1 | BOOL | 1 |
| In2 | Input In2 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#FF |
| In2MsgEn | 1 = Activate message for input In2 | BOOL | 1 |
| In3 | Input In3 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#FF |
| In3MsgEn | 1 = Activate message for input In3 | BOOL | 1 |
| In4 | Input In4 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#FF |
| In4MsgEn | 1 = Activate message for input In4 | BOOL | 1 |
| In5 | Input In5 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#FF |
| In5MsgEn | 1 = Activate message for input In5 | BOOL | 1 |
| In6 | Input In6 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#FF |
| In6MsgEn | 1 = Activate message for input In6 | BOOL | 1 |
| In7 | Input In7 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#FF |
| In7MsgEn | 1 = Activate message for input In7 | BOOL | 1 |
| In8 | Input In8 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#FF |
| In8MsgEn | 1 = Activate message for input In8 | BOOL | 1 |
| InvIn1 | 1 = Invert input In1 | BOOL | 0 |
| InvIn2 | 1 = Invert input In2 | BOOL | 0 |
| InvIn3 | 1 = Invert input In3 | BOOL | 0 |
| InvIn4 | 1 = Invert input In4 | BOOL | 0 |
| InvIn5 | 1 = Invert input In5 | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| InvIn6 | 1 = Invert input In6 | BOOL | 0 |
| InvIn7 | 1 = Invert input In7 | BOOL | 0 |
| InvIn8 | 1 = Invert input In8 | BOOL | 0 |
| MS_RelOp | 1 = Release for maintenance by OS operator | BOOL | 0 |
| MsgEvId | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>Refer also to the Suppressing messages using the MsgLock parameter (Page 231) section for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OS_Perm | I/O for operator control permissions (Page 2033) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | RESERVED | ANY | |
| SelFp2 | RESERVED | ANY | |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see EventNck error handling (Page 2035). | INT | -1 |
| MsgErr | 1 = Alarm error (output ERROR of NOTIFY_8P) | BOOL | 0 |
| MsgStat | Alarm status (output STATUS of NOTIFY_8P) | WORD | 16#0000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 2030) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 2030) | DWORD | 16#00000000 |

### See also

EventNck modes (Page 2032)

EventNck messaging (Page 2036)

EventNck block diagram (Page 2041)

Configurable functions using the Feature I/O (Page 139)

Opening additional faceplates (Page 234)

## 10.2.7 EventNck block diagram

### EventNck block diagram

A block diagram is not provided for this block.

### See also

Description of EventNck (Page 2030)

EventNck modes (Page 2032)

EventNck functions (Page 2033)

EventNck error handling (Page 2035)

EventNck messaging (Page 2036)

EventNck I/Os (Page 2037)

# 10.3 EventTs - Creating messages with time stamp

## 10.3.1 Description of EventTs

### Object name (type + number) and family

Type + number: FB 1812

Family: Report

### Area of application for EventTs

The block is used for the following applications:

- Generating messages which have to be acknowledged for time-stamped signals

### How it works

The block must be linked to a channel block and monitors up to eight different binary signals. From these, it generates time-stamped messages requiring acknowledgment that appear in the alarm view of the technological block to which it is connected.

The individual messages are assigned to the monitored signals via the inputs and the messages released or blocked depending on the process status data. If a change is made to one or more of the monitored signals enabled for reporting, a message is output containing a time stamp for the signal change.

`EventTs` can operate independently or in combination with one of the technological blocks that has the `EventTsIn` or `EventTs2In` input parameter. When this interconnection is configured, the messages of the `EventTs` block are displayed in the OS of the alarm view of the technological block and can also be acknowledged there. The following functions are controlled by the technological block:

- Switching to "Out of service" mode and switching back to "On" mode: `OosAct`, `OnAct`
- Release for maintenance: MS_Release
- Suppressing messages using the `MsgLock` parameter
- Batch parameters: `BatchEn`, `BatchID`, `BatchName`, `StepNo`, `Occupied`

**Note**

The `EventTs` instance must be newly created between deleting the interconnection of `EventTs` and the technological block and a delta download.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100) and in OB1 because of the time stamp.

If the EventTs operates together with a technological block, interconnect the output parameter EventTsOut of the EventTs block and the input parameter EventTsIn or EventTs2In of the technological block in the CFC.

Depending on how the time stamp is generated, the signals to be monitored for this are interconnected at inputs In1 ... In8 or InTS1 ... InTS8. Each Inx or InTSx signal can also be inverted via input Invx. A message is output if a signal value is changed (but taking inversion into account).

Each input is assigned a separate message text. A message received at input InTS5, for example, is output with the message text for the SIG 5 signal.

Configuration of the alarms for negative edges 1 → 0 in HW Config only effects the signal inputs InTS1 ... InTS8 of EventTs.

If TimeStrampOn = 0 , meaning the inputs In1 ... In8 are used, you will have to set the input Invx = 1 to generate an alarm 1 → 0 at input Inx for a negative edge.

**Note**

Interconnection of the block to multiple technological blocks is not permitted.

For the EventTs block, the Advanced Process Library contains process tag type templates; these serve as examples by providing various application scenarios for this block.

Examples of process tag types:

- Motor with an additional analog value and time-stamped signals (Motor_AV_EventTs) (Page 2863)

## Startup characteristics

During startup, the message block suppresses all messages. The duration (number of cycles) of message suppression is set at the RunUpCyc parameter. During restart (OB100) an internal counter that is initialized with this value decrements each time the block is executed. Messages are not generated if the counter value does not equal zero.

## Status word allocation for Status1 parameter

You can find a description for each parameter in section EventTs I/Os (Page 2052).

| Status bit | Parameter |
|---|---|
| 0 | Active signal 1 |
| 1 | Active signal 2 |
| 2 | Active signal 3 |
| 3 | Active signal 4 |
| 4 | Active signal 5 |
| 5 | Active signal 6 |
| 6 | Active signal 7 |
| 7 | Active signal 8 |
| 8 | InvIn1 |
| 9 | InvIn2 |

| Status bit | Parameter |
|---|---|
| 10 | InvIn3 |
| 11 | InvIn4 |
| 12 | InvIn5 |
| 13 | InvIn6 |
| 14 | InvIn7 |
| 15 | InvIn8 |
| 16 | Active signal 1 with inversion |
| 17 | Active signal 2 with inversion |
| 18 | Active signal 3 with inversion |
| 19 | Active signal 4 with inversion |
| 20 | Active signal 5 with inversion |
| 21 | Active signal 6 with inversion |
| 22 | Active signal 7 with inversion |
| 23 | Active signal 8 with inversion |
| 24 | Active signal 1 not connected |
| 25 | Active signal 2 not connected |
| 26 | Active signal 3 not connected |
| 27 | Active signal 4 not connected |
| 28 | Active signal 5 not connected |
| 29 | Active signal 6 not connected |
| 30 | Active signal 7 not connected |
| 31 | Active signal 8 not connected |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | In1MsgEn |
| 1 | In2MsgEn |
| 2 | In3MsgEn |
| 3 | In4MsgEn |
| 4 | In5MsgEn |
| 5 | In6MsgEn |
| 6 | In7MsgEn |
| 7 | In8MsgEn |
| 8 | Effective signal 1 for message |
| 9 | Effective signal 2 for message |
| 10 | Effective signal 3 for message |
| 11 | Effective signal 4 for message |
| 12 | Effective signal 5 for message |
| 13 | Effective signal 6 for message |
| 14 | Effective signal 7 for message |
| 15 | Effective signal 8 for message |

| Status bit | Parameter |
|---|---|
| 16 | MsgLock |
| 17 - 24 | Not used |
| 25 | Occupied |
| 26 | BatchEn |
| 27 | Batch - parameter exists |
| 28 - 31 | Not used |

### See also

EventTs block diagram (Page 2056)

EventTs messaging (Page 2049)

EventTs error handling (Page 2048)

EventTs functions (Page 2046)

EventTs modes (Page 2045)

## 10.3.2 EventTs modes

### EventTs operating modes

The block provides the following modes:

- On (Page 77)
- Out of service (Page 77)

### "On"

General information on the "On" mode is available in the section On (Page 77).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

### See also

EventTs block diagram (Page 2056)

EventTs I/Os (Page 2052)

EventTs messaging (Page 2049)

EventTs error handling (Page 2048)

EventTs functions (Page 2046)

Description of EventTs (Page 2042)

## 10.3.3 EventTs functions

### Functions of EventTs

The functions for this block are listed below.

### Activation and deactivation of messages

You can set the I/Os `InMsgEn1` ... `InMsgEn8` to individually enable or disable the messages applied to inputs `In1` ... `In8` or `InTS1` ... `InTS8`. All messages are activated by default.

To deactivate messages received at I/O `InTS4`, for example, you set I/O `InMsgEn4` = 0 accordingly.

You can deactivate all messages via I/O `MsgLock` = 1 .

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Time stamp as associated value of a message

You can use input `TimeStampOn` to select how the time stamp for the signals of EventTs will be generated:

- If you want to use the high-precision time stamp from the I/O devices, set `TimeStampOn` = 1. Interconnect one of the `InTSx` inputs with output `TS_Out` of block Pcs7DiIT.
- If you want to use the time stamp from the CPU, set `TimeStampOn` = 0. Interconnect one of the `Inx` inputs with output `PV_Out` of block Pcs7DiIT or with an appropriate output of a different block.

For additional time stamp properties, refer to the description of the standard function Time stamp (Page 231).

### Signal status as associated value of a message

The signal status as an associated value of the message is output for every signal, as is the time stamp.

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Operator permissions

The block has the following Operator control permissions (Page 285) for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 - 31 | Not used |

This block does not have a faceplate yet; the operator permissions have already been assigned in the planning phase for these faceplates.

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Configurable functions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 1 | Reaction to the out of service mode (Page 188) |
| 8 | Reporting with BATCH parameters (Page 167) |
| 9 | Enable external message (Page 200) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |

### Displaying and outputting the signal status

The block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The block determines the worst signal status via all interconnected binary inputs (according to `TimeStampOn`) and outputs this value at `ST_Worst`.

- `TimeStampOn` = 0
  - `In1`

  etc. to

  - `In8`
- `TimeStampOn` = 1
  - `InTS1`

  etc. to

  - `InTS8`

### See also

EventTs block diagram (Page 2056)

EventTs I/Os (Page 2052)

EventTs messaging (Page 2049)

EventTs error handling (Page 2048)

EventTs modes (Page 2045)

Description of EventTs (Page 2042)

## 10.3.4 EventTs error handling

### Error handling of EventTs

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |

### See also

EventTs block diagram (Page 2056)

EventTs I/Os (Page 2052)

EventTs messaging (Page 2049)

EventTs functions (Page 2046)

EventTs modes (Page 2045)

Description of EventTs (Page 2042)

## 10.3.5 EventTs messaging

### Messaging

Messages which can be acknowledged are generated using ALARM_8P. The ALARM_8P has 8 digital inputs and 10 associated values. Every edge transition detected for one or more digital inputs results in a message. The associated values are assigned consistently to the message at the time of edge evaluation. All 8 signals are assigned a common message number, which is split at the OS into 8 messages. The Engineering System (ES) assigns the message number automatically by calling the message server.

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId 1` | SIG 1 | AS process control message - fault | $$BlockComment$$ `InTS1` Status 16#@1%x@ |
| | SIG 2 | AS process control message - fault | $$BlockComment$$ `InTS2` Status 16#@2%x@ |
| | SIG 3 | AS process control message - fault | $$BlockComment$$ `InTS3` Status 16#@3%x@ |
| | SIG 4 | AS process control message - fault | $$BlockComment$$ `InTS1` Status 16#@4%x@ |
| | SIG 5 | AS process control message - fault | $$BlockComment$$ `InTS5` Status 16#@5%x@ |
| | SIG 6 | AS process control message - fault | $$BlockComment$$ `InTS6` Status 16#@6%x@ |
| | SIG 7 | AS process control message - fault | $$BlockComment$$ `InTS7` Status 16#@7%x@ |
| | SIG 8 | AS process control message - fault | $$BlockComment$$ `InTS8` Status 16#@8%x@ |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

The following applies for `TimeStampOn` = 0 : 16#@n%x@ (n = 1 ... 8): The value contains the signal status of `In1` ... `In8`

The following applies for `TimeStampOn` = 1 : 16#@n%x@ (n = 1 ... 8): Statement on the validity of the time stamp from `InTS0` ... `InTS8`. If the value is 80, the time stamp is formed by the I/O. If the value is ≠ 80, the time stamp of the I/O is invalid; the time stamp is generated by the CPU as a substitute and is therefore not exact.

You can change the message class and the event to meet your needs at the block type and/or block instance.

### Associated values for message instance **MsgEvId**

| Associated value | Block parameters |
|---|---|
| 1 | `TimeStampOn` = 0: `In1.ST` |
| 2 | `TimeStampOn` = 0: `In2.ST` |
| 3 | `TimeStampOn` = 0: `In3.ST` |
| 4 | `TimeStampOn` = 0: `In4.ST` |
| 5 | `TimeStampOn` = 0: `In5.ST` |
| 6 | `TimeStampOn` = 0: `In6.ST` |
| 7 | `TimeStampOn` = 0: `In7.ST` |
| 8 | `TimeStampOn` = 0: `In8.ST` |
| 9 | Not allocated |
| 10 | Not allocated |

### Associated values for message instance **MsgEvId**

| Associated value | Block parameters |
|---|---|
| 1 | `TimeStampOn` = 1: `InTS1.ST` |
| 2 | `TimeStampOn` = 1: `InTS2.ST` |
| 3 | `TimeStampOn` = 1: `InTS3.ST` |
| 4 | `TimeStampOn` = 1: `InTS4.ST` |
| 5 | `TimeStampOn` = 1: `InTS5.ST` |
| 6 | `TimeStampOn` = 1: `InTS6.ST` |
| 7 | `TimeStampOn` = 1: `InTS7.ST` |
| 8 | `TimeStampOn` = 1: `InTS8.ST` |
| 9 | Not allocated |
| 10 | Not allocated |

### The batch information is transmitted with `Feature.Bit8` =1:

The first three associated values are described as follows and are followed by the signal status of the input signals or the validity of the time stamp depending on `TimeStampOn`:

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchId |
| 4 | In1.ST / InTS1.ST |
| 5 | In2.ST / InTS2.ST |
| 6 | In3.ST / InTS3.ST |
| 7 | In4.ST / InTS4.ST |
| 8 | In5.ST / InTS5.ST |
| 9 | In6.ST / InTS6.ST |

The tenth associated value is not available.

Enter the batch ID @1%s@ under "Properties - Block - Special properties - Messages extended - Message texts block".

The associated values in the process messages (event) must be incremented by 3:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId 1` | SIG 1 | AS process control message - fault | $$BlockComment$$ `InTS1` Status 16#@4%x@ |
| | SIG 2 | AS process control message - fault | $$BlockComment$$ `InTS2` Status 16#@5%x@ |
| | SIG 3 | AS process control message - fault | $$BlockComment$$ `InTS3` Status 16#@6%x@ |
| | SIG 4 | AS process control message - fault | $$BlockComment$$ `InTS1` Status 16#@7%x@ |
| | SIG 5 | AS process control message - fault | $$BlockComment$$ `InTS5` Status 16#@8%x@ |
| | SIG 6 | AS process control message - fault | $$BlockComment$$ `InTS6` Status 16#@9%x@ |
| | SIG 7 | AS process control message - fault | $$BlockComment$$ `InTS7` Status[1] |
| | SIG 8 | AS process control message - fault | $$BlockComment$$ `InTS8` Status[1] |

[1] You cannot specify a separate associated value here.

## See also

EventTs block diagram (Page 2056)

EventTs I/Os (Page 2052)

EventTs error handling (Page 2048)

EventTs functions (Page 2046)

EventTs modes (Page 2045)

Description of EventTs (Page 2042)

Selecting values associated with messages (Page 167)

## 10.3.6 EventTs I/Os

I/Os of EventTs

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Inputs

| Parameter | Description | Type | Default |
|---|---|---|---|
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00 |
| BatchName | Batch name | S7-string | |
| EN | 1 = Called block will be processed | BOOL | 1 |
| ExtVal01 | External message 1 | ANY | |
| ExtVal02 | External message 2 | ANY | |
| ExtVal03 | External message 3 | ANY | |
| Feature | I/O for additional functions (Page 2046) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| In1 | Signal 1 for CPU time stamp | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In2 | Signal 2 for CPU time stamp | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In3 | Signal 3 for CPU time stamp | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In4 | Signal 4 for CPU time stamp | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In5 | Signal 5 for CPU time stamp | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| In6 | Signal 6 for CPU time stamp | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In7 | Signal 7 for CPU time stamp | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| In8 | Signal 8 for CPU time stamp | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| InTS1 | Signal 1 with I/O-device time stamp | ANY | |
| InTS2 | Signal 2 with I/O-device time stamp | ANY | |
| InTS3 | Signal 3 with I/O-device time stamp | ANY | |
| InTS4 | Signal 4 with I/O-device time stamp | ANY | |
| InTS5 | Signal 5 with I/O-device time stamp | ANY | |
| InTS6 | Signal 6 with I/O-device time stamp | ANY | |
| InTS7 | Signal 7 with I/O-device time stamp | ANY | |
| InTS8 | Signal 8 with I/O-device time stamp | ANY | |
| Inv1 | 1 = Invert input In1 or InTS1 | BOOL | 0 |
| Inv2 | 1 = Invert input In2 or InTS2 | BOOL | 0 |
| Inv3 | 1 = Invert input In3 or InTS3 | BOOL | 0 |
| Inv4 | 1 = Invert input In4 or InTS4 | BOOL | 0 |
| Inv5 | 1 = Invert input In5 or InTS5 | BOOL | 0 |
| Inv6 | 1 = Invert input In6 or InTS6 | BOOL | 0 |
| Inv7 | 1 = Invert input In7 or InTS7 | BOOL | 0 |
| Inv8 | 1 = Invert input In8 or InTS8 | BOOL | 0 |
| MsgEn1 | 1 = Activate message for input In1 or InTS1 | BOOL | 1 |
| MsgEn2 | 1 = Activate message for input In2 or InTS2 | BOOL | 1 |
| MsgEn3 | 1 = Activate message for input In3 or InTS3 | BOOL | 1 |
| MsgEn4 | 1 = Activate message for input In4 or InTS4 | BOOL | 1 |
| MsgEn5 | 1 = Activate message for input In5 or InTS5 | BOOL | 1 |
| MsgEn6 | 1 = Activate message for input In6 or InTS6 | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MsgEn7 | 1 = Activate message for input In7 or InTS7 | BOOL | 1 |
| MsgEn8 | 1 = Activate message for input In8 or InTS8 | BOOL | 1 |
| MsgEvId | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_RelOp | 1 = Release for maintenance by OS operator | BOOL | 0 |
| Occupied | 1 = In use by a batch | BOOL | 0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OS_Perm | I/O for EventTs functions (Page 2046) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | RESERVED | ANY | |
| SelFp2 | RESERVED | ANY | |
| StepNo | Batch step number | DWORD | 16#00 |
| TimeStampOn | 0 = Use time stamp of the CPU<br>1 = Use time stamp of the I/O devices | BOOL | 0 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Outputs

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see EventTs error handling (Page 2048) | INT | -1 |
| EventTsOut | For maneuvering of data between a technological block and the message blocks EventTs, Event16Ts. | STRUCT | |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn | Message acknowledgment status (output ACK_STATE of ALARM_8P) | WORD | 16#0000 |
| MsgErr | 1 = Message error (output ERROR of ALARM_8P) | BOOL | 0 |
| MsgStat | Message status (output STATUS of ALARM_8P) | WORD | 16#0000 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 2042) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 2042) | DWORD | 16#00000000 |

## See also

EventTs block diagram (Page 2056)

EventTs messaging (Page 2049)

EventTs modes (Page 2045)

Opening additional faceplates (Page 234)

## 10.3.7 EventTs block diagram

### EventTs block diagram

A block diagram is not provided for this block.

### See also

EventTs I/Os (Page 2052)
EventTs messaging (Page 2049)
EventTs error handling (Page 2048)
EventTs functions (Page 2046)
EventTs modes (Page 2045)
Description of EventTs (Page 2042)

# 10.4 Event16Ts - Creating 16 messages with time stamp

## 10.4.1 Description of Event16Ts

### Object name (type + number) and family

Type + number: FB 1887

Family: Report

### Area of application for Event16Ts

The block is used for the following applications:

- Generating messages which have to be acknowledged for time-stamped signals

For Event16Ts block, use the templates for the process tag types of the EventTs block in the Advanced Process Library as examples with various use cases. Replace the EventTs block with the Event16Ts block.

### How it works

The block monitors up to sixteen different binary signals, connect the block to a channel block.

The block generates time-stamped messages requiring acknowledgment that appear in the alarm view of the technological block to which it is connected.

The individual messages are assigned to the monitored signals via the inputs and the messages released or blocked depending on the process status data. If a change is made to one or more of the monitored signals enabled for reporting, a message is output containing a time stamp for the signal change.

`Event16Ts` can operate independently or in combination with one of the technological blocks that has the `EventTsIn` or `EventTs2In` input parameter. When the interconnection is configured, the messages of the `Event16Ts` block are displayed in the OS of the alarm view of the technological block and you can also acknowledge the messages. The following functions are controlled by the technological block:

- Switching to "Out of service" mode and switching back to "On" mode: `OosAct`, `OnAct`
- Release for maintenance: MS_Release
- Suppressing messages using the `MsgLock` parameter
- Batch parameters: `BatchEn`, `BatchID`, `BatchName`, `StepNo`, `Occupied`

**Note**

The `Event16Ts` instance must be newly created between deleting the interconnection of `Event16Ts` and the technological block and a delta download.

## Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100) and in OB1 because for time stamping.

If the `Event16Ts` operates together with a technological block, interconnect the output parameter `EventTsOut` of the `Event16Ts` block and the input parameter `EventTsIn` or `EventTs2In` of the technological block in the CFC.

Depending on how the time stamp is generated, the signals to be monitored for this are interconnected at inputs `In1` ... `In16` or `InTS1` ... `InTS16`. Each `Inx` or `InTSx` signal can also be inverted via input `Invx`. If a signal value changes taking inversion into account, a message is output.

**Note**

1. The input parameters `ExtVal01`, `ExtVal02`, and `ExtVal03` will be used for `In1`...`In8` and `InTS1`...`InTS8`.
2. The input parameters `ExtVal04`, `ExtVal05`, and `ExtVal06` will be used for `In9`...`In16` and `InTS9`...`InTS16`.

Each input is assigned a separate message text. A message received at input `InTS5`, for example, is output with the message text for the SIG 5 signal.

Configuration of the alarms for negative edges 1 → 0 in HW Config only effects the signal inputs `InTS1` ... `InTS16` of Event16Ts.

If `TimeStrampOn` = 0 , meaning the inputs `In1` ... `In16` are used, you will have to set the input `Invx` = 1 to generate an alarm 1 → 0 at input `Inx` for a negative edge.

**Note**

Interconnection of the block to multiple technological blocks is not permitted.

For the Event16Ts block, the Advanced Process Library contains templates for process tag types as an example with various application scenarios for this block.

Examples of process tag types:

- Motor with an additional analog value and time-stamped signals (Motor_AV_EventTs) (Page 2863)

## Startup characteristics

During startup, the message block suppresses all messages. The duration (number of cycles) of message suppression is set at the `RunUpCyc` parameter. During restart (OB100) an internal counter that is initialized with this value decrements each time the block is executed. Messages are not generated if the counter value does not equal zero.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section I/Os of Event16Ts.

| Status bit | Parameter |
|---|---|
| 0 | Active signal 1 with inversion |
| 1 | Active signal 2 with inversion |
| 2 | Active signal 3 with inversion |
| 3 | Active signal 4 with inversion |
| 4 | Active signal 5 with inversion |
| 5 | Active signal 6 with inversion |
| 6 | Active signal 7 with inversion |
| 7 | Active signal 8 with inversion |
| 8 | Active signal 9 with inversion |
| 9 | Active signal 10 with inversion |
| 10 | Active signal 11 with inversion |
| 11 | Active signal 12 with inversion |
| 12 | Active signal 13 with inversion |
| 13 | Active signal 14 with inversion |
| 14 | Active signal 15 with inversion |
| 15 | Active signal 16 with inversion |
| 16 | Active signal 1 not connected |
| 17 | Active signal 2 not connected |
| 18 | Active signal 3 not connected |
| 19 | Active signal 4 not connected |
| 20 | Active signal 5 not connected |
| 21 | Active signal 6 not connected |
| 22 | Active signal 7 not connected |
| 23 | Active signal 8 not connected |
| 24 | Active signal 9 not connected |
| 25 | Active signal 10 not connected |
| 26 | Active signal 11 not connected |
| 27 | Active signal 12 not connected |
| 28 | Active signal 13 not connected |
| 29 | Active signal 14 not connected |
| 30 | Active signal 15 not connected |
| 31 | Active signal 16 not connected |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | Active signal 1 |
| 1 | Active signal 2 |
| 2 | Active signal 3 |

| Status bit | Parameter |
|---|---|
| 3 | Active signal 4 |
| 4 | Active signal 5 |
| 5 | Active signal 6 |
| 6 | Active signal 7 |
| 7 | Active signal 8 |
| 8 | Active signal 9 |
| 9 | Active signal 10 |
| 10 | Active signal 11 |
| 11 | Active signal 12 |
| 12 | Active signal 13 |
| 13 | Active signal 14 |
| 14 | Active signal 15 |
| 15 | Active signal 16 |
| 16 | InvIn1 |
| 17 | InvIn2 |
| 18 | InvIn3 |
| 19 | InvIn4 |
| 20 | InvIn5 |
| 21 | InvIn6 |
| 22 | InvIn7 |
| 23 | InvIn8 |
| 24 | InvIn9 |
| 25 | InvIn10 |
| 26 | InvIn11 |
| 27 | InvIn12 |
| 28 | InvIn13 |
| 29 | InvIn14 |
| 30 | InvIn15 |
| 31 | InvIn16 |

### Status word allocation for `Status3` parameter

| Status bit | Parameter |
|---|---|
| 0 | In1MsgEn |
| 1 | In2MsgEn |
| 2 | In3MsgEn |
| 3 | In4MsgEn |
| 4 | In5MsgEn |
| 5 | In6MsgEn |
| 6 | In7MsgEn |
| 7 | In8MsgEn |
| 8 | In9MsgEn |

| Status bit | Parameter |
|---|---|
| 9 | In10MsgEn |
| 10 | In11MsgEn |
| 11 | In12MsgEn |
| 12 | In13MsgEn |
| 13 | In14MsgEn |
| 14 | In15MsgEn |
| 15 | In16MsgEn |
| 16 | Effective signal 1 for message |
| 17 | Effective signal 2 for message |
| 18 | Effective signal 3 for message |
| 19 | Effective signal 4 for message |
| 20 | Effective signal 5 for message |
| 21 | Effective signal 6 for message |
| 22 | Effective signal 7 for message |
| 23 | Effective signal 8 for message |
| 24 | Effective signal 9 for message |
| 25 | Effective signal 10 for message |
| 26 | Effective signal 11 for message |
| 27 | Effective signal 12 for message |
| 28 | Effective signal 13 for message |
| 29 | Effective signal 14 for message |
| 30 | Effective signal 15 for message |
| 31 | Effective signal 16 for message |

### Status word allocation for `Status4` parameter

| Status bit | Parameter |
|---|---|
| 0-7 | Not used |
| 8 - 15 | Not used |
| 16 | `MsgLock` |
| 7 - 24 | Not used |
| 25 | `Occupied` |
| 26 | `BatchEn` |
| 27 | Batch - parameter exists |
| 28 - 31 | Not used |

### See also

Event16Ts block diagram (Page 2073)
Event16Ts I/Os (Page 2071)
Event16Ts messages (Page 2065)
Event16Ts error handling (Page 2065)

Event16Ts functions (Page 2062)
Event16Ts operating modes (Page 2062)

## 10.4.2 Event16Ts operating modes

### Event16Ts operating modes

The block provides the following modes:

- On (Page 77)
- Out of service (Page 77)

### "On"

General information on the "On" mode is available in the section On (Page 77).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

### See also

Event16Ts block diagram (Page 2073)
Event16Ts I/Os (Page 2071)
Event16Ts messages (Page 2065)
Event16Ts error handling (Page 2065)
Event16Ts functions (Page 2062)
Description of Event16Ts (Page 2057)

## 10.4.3 Event16Ts functions

### Functions of Event16Ts

The functions for this block are listed below.

### Activation and deactivation of messages

You can set the I/Os `InMsgEn1` ... `InMsgEn8` to individually enable or disable the messages applied to inputs `In1` ... `In8` or `InTS1` ... `InTS8`. All messages are activated by default.

To deactivate messages received at I/O `InTS4`, for example, you set I/O `InMsgEn4` = 0 accordingly.

You can deactivate all messages via I/O `MsgLock` = 1 .

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Time stamp as associated value of a message

You can use input `TimeStampOn` to select how the time stamp for the signals of Event16Ts will be generated:

- If you want to use the high-precision time stamp from the I/O devices, set `TimeStampOn` = 1. Interconnect one of the `InTSx` inputs with output `TS_Out` of block Pcs7DiIT.
- If you want to use the time stamp from the CPU, set `TimeStampOn` = 0. Interconnect one of the `Inx` inputs with output `PV_Out` of block Pcs7DiIT or with an appropriate output of a different block.

For additional time stamp properties, refer to the description of the standard function Time stamp (Page 231).

### Signal status as associated value of a message

The signal status as an associated value of the message is output for every signal, as is the time stamp.

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Operator permissions

The block has the following Operator control permissions (Page 285) for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 - 31 | Not used |

This block does not have a faceplate yet; the operator permissions have already been assigned in the planning phase for these faceplates.

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Configurable functions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 1 | Reaction to the out of service mode (Page 188) |
| 8 | Reporting with BATCH parameters (Page 167) |
| 9 | Enable external message (Page 200) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |

### Displaying and outputting the signal status

The block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The block determines the worst signal status via all interconnected binary inputs (according to `TimeStampOn`) and outputs this value at `ST_Worst`.

- `TimeStampOn` = 0
  - `In1`

  etc. to

  - `In16`
- `TimeStampOn` = 1
  - `InTS1`

  etc. to

  - `InTS16`

### See also

Event16Ts block diagram (Page 2073)

Event16Ts I/Os (Page 2071)

Event16Ts messages (Page 2065)

Event16Ts error handling (Page 2065)

Event16Ts operating modes (Page 2062)
Description of Event16Ts (Page 2057)

## 10.4.4 Event16Ts error handling

### Error handling of EventTs

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed. |
| 0 | There is no error. |

### See also

Event16Ts block diagram (Page 2073)
Event16Ts I/Os (Page 2071)
Event16Ts messages (Page 2065)
Event16Ts functions (Page 2062)
Event16Ts operating modes (Page 2062)
Description of Event16Ts (Page 2057)

## 10.4.5 Event16Ts messages

### Messaging

Messages which can be acknowledged are generated using ALARM_8P. The ALARM_8P has 8 digital inputs and 10 associated values. Every edge transition detected for one or more digital inputs results in a message. The associated values are assigned consistently to the message at the time of edge evaluation. All 8 signals are assigned a common message number, which is split at the OS into 8 messages. The Engineering System (ES) assigns the message number automatically by calling the message server.

### Process messages (MsgEvId1)

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId1` | SIG 1 | AS process control message - fault | $$BlockComment$$ `InTS1` Status 16#@1%x@ |
| | SIG 2 | AS process control message - fault | $$BlockComment$$ `InTS2` Status 16#@2%x@ |
| | SIG 3 | AS process control message - fault | $$BlockComment$$ `InTS3` Status 16#@3%x@ |
| | SIG 4 | AS process control message - fault | $$BlockComment$$ `InTS4` Status 16#@4%x@ |
| | SIG 5 | AS process control message - fault | $$BlockComment$$ `InTS5` Status 16#@5%x@ |
| | SIG 6 | AS process control message - fault | $$BlockComment$$ `InTS6` Status 16#@6%x@ |
| | SIG 7 | AS process control message - fault | $$BlockComment$$ `InTS7` Status 16#@7%x@ |
| | SIG 8 | AS process control message - fault | $$BlockComment$$ `InTS8` Status 16#@8%x@ |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

The following applies for `TimeStampOn` = 0 : 16#@n%x@ (n = 1 ... 8): The value contains the signal status of `In1` ... `In8`

The following applies for `TimeStampOn` = 1 : 16#@n%x@ (n = 1 ... 8): Statement on the validity of the time stamp from `InTS0` ... `InTS8`. If the value is 80, the time stamp is formed by the I/O. If the value is ≠ 80, the time stamp of the I/O is invalid; the time stamp is generated by the CPU as a substitute and is therefore not exact.

You can change the message class and the event to meet your needs at the block type and/or block instance.

### Process messages (MsgEvId2)

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId2 | SIG 1 | AS process control message - fault | $$BlockComment$$ InTS9 Status 16#@1%x@ |
| | SIG 2 | AS process control message - fault | $$BlockComment$$ InTS10 Status 16#@2%x@ |
| | SIG 3 | AS process control message - fault | $$BlockComment$$ InTS11 Status 16#@3%x@ |
| | SIG 4 | AS process control message - fault | $$BlockComment$$ InTS12 Status 16#@4%x@ |
| | SIG 5 | AS process control message - fault | $$BlockComment$$ InTS13 Status 16#@5%x@ |
| | SIG 6 | AS process control message - fault | $$BlockComment$$ InTS14 Status 16#@6%x@ |
| | SIG 7 | AS process control message - fault | $$BlockComment$$ InTS15 Status 16#@7%x@ |
| | SIG 8 | AS process control message - fault | $$BlockComment$$ InTS16 Status 16#@8%x@ |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

The following applies for TimeStampOn = 0 : 16#@n%x@ (n = 1 ... 8): The value contains the signal status of In9 ... In16

The following applies for TimeStampOn = 1 : 16#@n%x@ (n = 1 ... 8): Statement on the validity of the time stamp from InTS9 ... InTS16. If the value is 80, the time stamp is formed by the I/O. If the value is ≠ 80, the time stamp of the I/O is invalid; the time stamp is generated by the CPU as a substitute and is therefore not exact.

You can change the message class and the event to meet your needs at the block type and/or block instance.

### Associated values for message instance MsgEvId1

| Associated value | Block parameters |
|---|---|
| 1 | TimeStampOn = 0: In1.ST |
| 2 | TimeStampOn = 0: In2.ST |
| 3 | TimeStampOn = 0: In3.ST |
| 4 | TimeStampOn = 0: In4.ST |
| 5 | TimeStampOn = 0: In5.ST |
| 6 | TimeStampOn = 0: In6.ST |
| 7 | TimeStampOn = 0: In7.ST |
| 8 | TimeStampOn = 0: In8.ST |
| 9 | Not allocated |
| 10 | Not allocated |

### Associated values for message instance **MsgEvId2**

| Associated value | Block parameters |
|---|---|
| 1 | TimeStampOn = 1: In9.ST |
| 2 | TimeStampOn = 1: In10.ST |
| 3 | TimeStampOn = 1: In11.ST |
| 4 | TimeStampOn = 1: In12.ST |
| 5 | TimeStampOn = 1: In13.ST |
| 6 | TimeStampOn = 1: In14.ST |
| 7 | TimeStampOn = 1: In15.ST |
| 8 | TimeStampOn = 1: In16.ST |
| 9 | Not allocated |
| 10 | Not allocated |

### Associated values for message instance **MsgEvId1**

| Associated value | Block parameters |
|---|---|
| 1 | TimeStampOn = 0: InTS1.ST |
| 2 | TimeStampOn = 0: InTS2.ST |
| 3 | TimeStampOn = 0: InTS3.ST |
| 4 | TimeStampOn = 0: InTS4.ST |
| 5 | TimeStampOn = 0: InTS5.ST |
| 6 | TimeStampOn = 0: InTS6.ST |
| 7 | TimeStampOn = 0: InTS7.ST |
| 8 | TimeStampOn = 0: InTS8.ST |
| 9 | Not allocated |
| 10 | Not allocated |

### Associated values for message instance **MsgEvId2**

| Associated value | Block parameters |
|---|---|
| 1 | TimeStampOn = 0: InTS9.ST |
| 2 | TimeStampOn = 0: InTS10.ST |
| 3 | TimeStampOn = 0: InTS11.ST |
| 4 | TimeStampOn = 0: InTS12.ST |
| 5 | TimeStampOn = 0: InTS13.ST |
| 6 | TimeStampOn = 0: InTS614.ST |
| 7 | TimeStampOn = 0: InTS15.ST |
| 8 | TimeStampOn = 0: InTS16.ST |
| 9 | Not allocated |
| 10 | Not allocated |

### The batch information is transmitted with `Feature.Bit8` =1:

The first three associated values are described as follows and are followed by the signal status of the input signals or the validity of the time stamp depending on `TimeStampOn`:

**MsgEvId1**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchId |
| 4 | In1.ST / InTS1.ST |
| 5 | In2.ST / InTS2.ST |
| 6 | In3.ST / InTS3.ST |
| 7 | In4.ST / InTS4.ST |
| 8 | In5.ST / InTS5.ST |
| 9 | In6.ST / InTS6.ST |

**MsgEvId2**

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchId |
| 4 | In7.ST / InTS7.ST |
| 5 | In8ST / InTS8.ST8 |
| 6 | In9.ST / InTS9.ST |
| 7 | In10.ST / InTS10.ST |
| 8 | In11.ST / InTS11.ST |
| 9 | I12.ST / InTS12.ST |

The tenth associated value is not available.

Enter the batch ID @1%s@ under "Properties - Block - Special properties - Messages extended - Message texts block".

The associated values in the process messages (event) must be incremented by 3:

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId1` | SIG 1 | AS process control message - fault | $$BlockComment$$ `InTS1` Status 16#@4%x@ |
| | SIG 2 | AS process control message - fault | $$BlockComment$$ `InTS2` Status 16#@5%x@ |
| | SIG 3 | AS process control message - fault | $$BlockComment$$ `InTS3` Status 16#@6%x@ |
| | SIG 4 | AS process control message - fault | $$BlockComment$$ `InTS4` Status 16#@7%x@ |
| | SIG 5 | AS process control message - fault | $$BlockComment$$ `InTS5` Status 16#@8%x@ |
| | SIG 6 | AS process control message - fault | $$BlockComment$$ `InTS6` Status 16#@9%x@ |
| | SIG 7 | AS process control message - fault | $$BlockComment$$ `InTS7` Status[1] |
| | SIG 8 | AS process control message - fault | $$BlockComment$$ `InTS8` Status[1] |

[1] You cannot specify a separate associated value here.

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId2` | SIG 1 | AS process control message - fault | $$BlockComment$$ `InTS9` Status 16#@4%x@ |
| | SIG 2 | AS process control message - fault | $$BlockComment$$ `InTS10` Status 16#@5%x@ |
| | SIG 3 | AS process control message - fault | $$BlockComment$$ `InTS11` Status 16#@6%x@ |
| | SIG 4 | AS process control message - fault | $$BlockComment$$ `InTS12` Status 16#@7%x@ |
| | SIG 5 | AS process control message - fault | $$BlockComment$$ `InTS13` Status 16#@8%x@ |
| | SIG 6 | AS process control message - fault | $$BlockComment$$ `InTS14` Status 16#@9%x@ |
| | SIG 7 | AS process control message - fault | $$BlockComment$$ `InTS15` Status[1] |
| | SIG 8 | AS process control message - fault | $$BlockComment$$ `InTS16` Status[1] |

[1] You cannot specify a separate associated value here.

### See also

Event16Ts block diagram (Page 2073)
Event16Ts I/Os (Page 2071)
Event16Ts error handling (Page 2065)
Event16Ts functions (Page 2062)
Event16Ts operating modes (Page 2062)
Description of Event16Ts (Page 2057)
Selecting values associated with messages (Page 167)

## 10.4.6 Event16Ts I/Os

I/Os of EventTs

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Inputs

| Parameter | Description | Type | Default |
|---|---|---|---|
| `BatchEn` | 1 = Enable allocation | BOOL | 0 |
| `BatchID` | Batch ID | DWORD | 16#00000000 |
| `BatchName` | Batch name | S7-string | |
| `EN` | 1 = Called block will be processed | BOOL | 1 |
| `Feature` | I/O for additional functions (Page 2062) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| `ExtVal01` | External message 1 | ANY | |
| `ExtVal02` | External message 2 | ANY | |
| `ExtVal03` | External message 3 | ANY | |
| `ExtVal04` | External message 4 | ANY | |
| `ExtVal05` | External message 5 | ANY | |
| `ExtVal06` | External message 6 | ANY | |
| `In1..In16` | Signal 1..16 for CPU time stamp | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#FF |
| `InTS1..InTS16` | Signal 1..16 with I/O-device time stamp | ANY | |
| `Inv1..Inc16` | 1 = Invert input `In1..In16` or `InTS1..InTS16` | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| `MsgEn1..MsgEn16` | 1 = Activate message for input `In1..In16` or `InTS1..InTS16` | BOOL | 1 |
| `MsgEvId1` | Message number (assigned automatically) | DWORD | 16#00000000 |
| `MsgEvId2` | Message number (assigned automatically) | DWORD | 16#00000000 |
| `MsgLock` | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `MS_RelOp` | 1 = Release for maintenance by OS operator | BOOL | 0 |
| `Occupied` | 1 = In use by a batch | BOOL | 0 |
| `OnOp*` | 1 = "On" mode via operator | BOOL | 0 |
| `OosLi` | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `OosOp*` | 1 = "Out of service", via OS operator | BOOL | 0 |
| `OS_Perm` | I/O for Event16Ts functions (Page 2062) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| `RunUpCyc` | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| `SampleTime` | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| `SelFp1` | RESERVED | ANY | |
| `SelFp2` | RESERVED | ANY | |
| `StepNo` | Batch step number | DWORD | 16#00000000 |
| `TimeStampOn` | 0 = Use time stamp of the CPU<br>1 = Use time stamp of the I/O devices | BOOL | 0 |
| `UserStatus` | First freely assignable bits for use in OS | BYTE | 16#00000000 |
| `UserStatus2` | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Outputs

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see Event16Ts error handling (Page 2065) | INT | -1 |
| EventTsOut | For maneuvering of data between a technological block and the message blocks EventTs, Event16Ts. | STRUCT | |
| MS_Release | Release for maintenance:<br>1 = Release for OS operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn1..MsgAckn2 | Message acknowledgment status (output ACK_STATE of ALARM_8P) | WORD | 16#0000 |
| MsgErr1..MsgErr2 | 1 = Message error (output ERROR of ALARM_8P) | BOOL | 0 |
| MsgStat1..MsgStat2 | Message status (output STATUS of ALARM_8P) | WORD | 16#0000 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 2057) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 2057) | DWORD | 16#00000000 |
| Status3 | Status word 3 (Page 2062) | DWORD | 16#00000000 |
| Status4 | Status word 4 (Page 2062) | DWORD | 16#00000000 |

### See also

Event16Ts block diagram (Page 2073)

Event16Ts messages (Page 2065)

Opening additional faceplates (Page 234)

## 10.4.7 Event16Ts block diagram

### EventTs block diagram

A block diagram is not provided for this block.

**See also**

Event16Ts I/Os (Page 2071)
Event16Ts messages (Page 2065)
Event16Ts error handling (Page 2065)
Event16Ts functions (Page 2062)
Event16Ts operating modes (Page 2062)
Description of Event16Ts (Page 2057)

# Counter blocks 11

## 11.1 CountScL - Counter with up and down counting direction

### 11.1.1 Description of CountScL

**Object name (type + number) and family**

Type + number: FB 1806

Family: Count

**Area of application for CountScL**

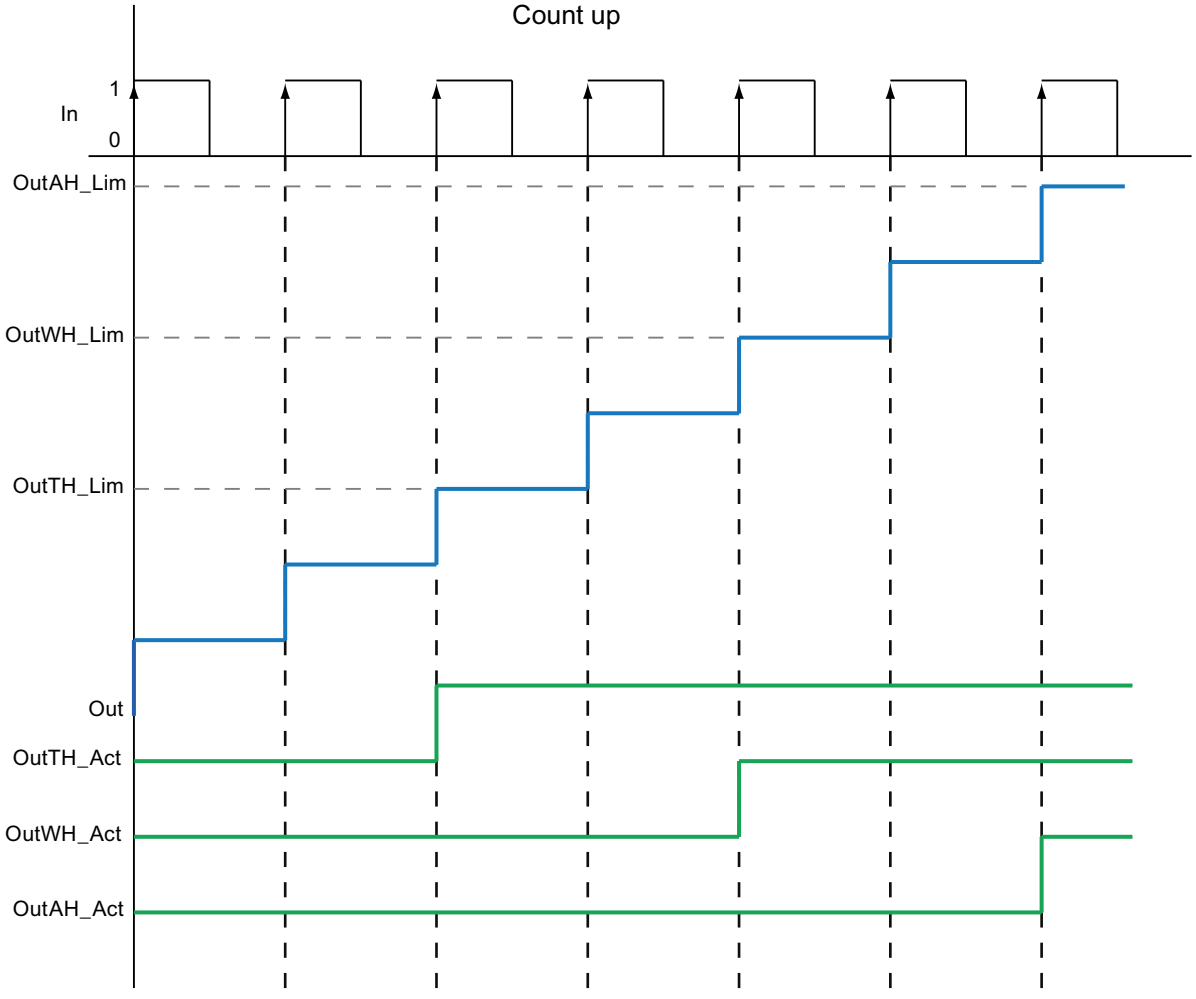
The block is used for the following applications:

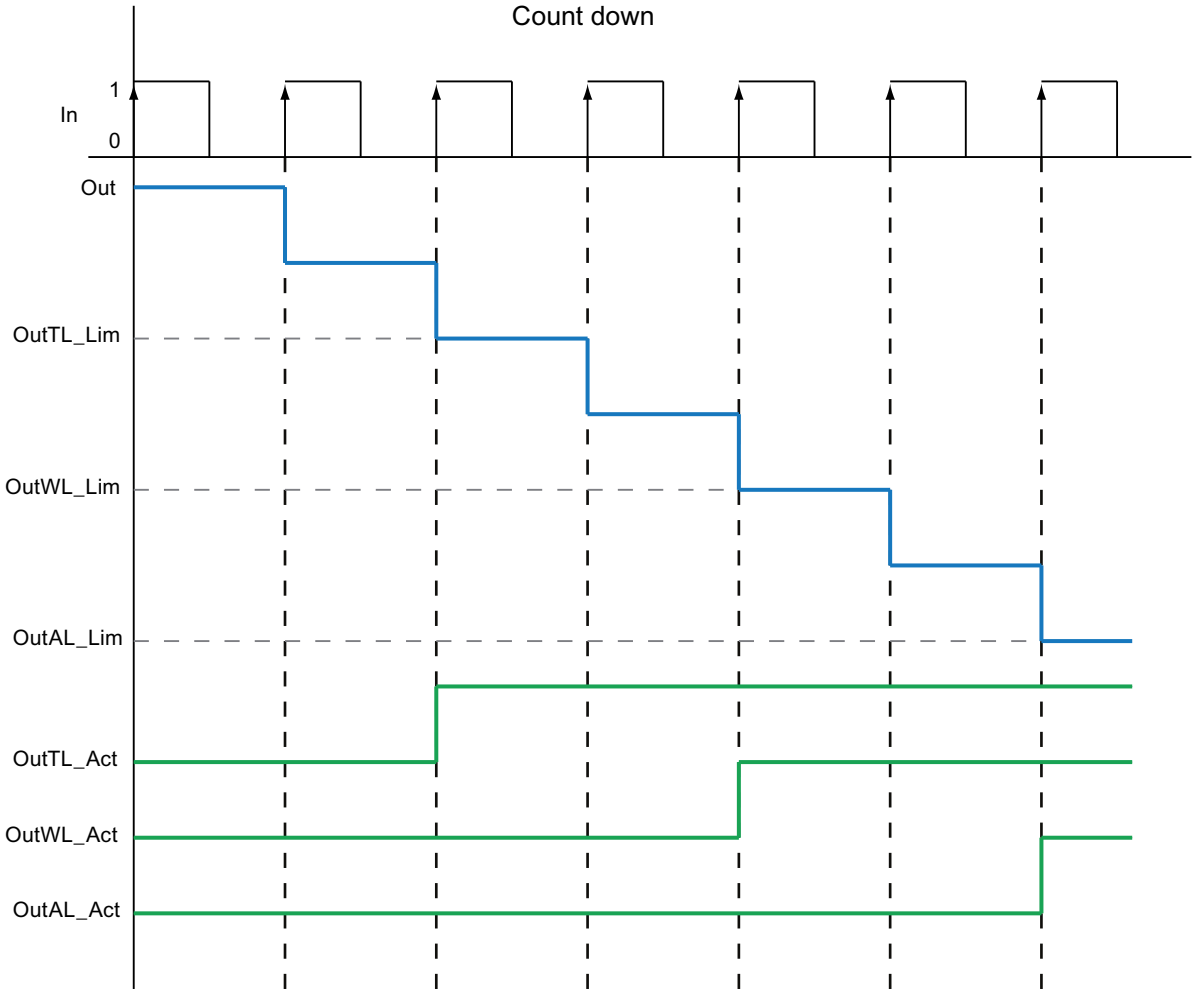
- Count up or count down with rising edge of the binary input signal.

## How it works

Positive edges at the binary input signal In increment or decrement the count value at Out, depending on settings.

1. Count up (`UpOp` = 1 or `UpLi` =1)
   The block counts up for each rising edge of `In`. (Output parameter `CountMode` = 1)
2. Count down (`DnOp` = 1 or `DnLi` = 1)
   The block counts down for each rising edge of `In`. (Output parameter `CountMode` = 2)
3. Off (`OffOp` = 1 or `OffLi` = 1)
   The block is disabled (output parameter `CountMode` = 0). No counting takes place.






## Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

## Startup characteristics

Define the startup characteristics for this block via two `Feature` Bits:

Bit 0: Set startup characteristics (Page 148)

Bit 5: Use the last value following a complete download as the current value during startup of the block (Page 165)

## Time response

The block does not have any time response.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section CountScL I/Os (Page 2085).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | Not used |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | Not used |
| 6 | `OnAct.Value` |
| 7 - 10 | Not used |
| 11 | `LiOp` |
| 12 - 13 | Not used |
| 14 | 1 = Invalid signal status |
| 15 | Not used |
| 16 | Internal error |
| 17-31 | Not used |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `MsgLock.Value` |
| 1 | `OutAH_Act.Value` |
| 2 | `OutWH_Act.Value` |
| 3 | `OutTH_Act.Value` |
| 4 | `OutTL_Act.Value` |
| 5 | `OutWL_Act.Value` |
| 6 | `OutAL_Act.Value` |
| 7 | `OutAH_En` OR `Feature.Bit23` |
| 8 | `OutWH_En` OR `Feature.Bit23` |
| 9 | `OutTH_En` OR `Feature.Bit23` |
| 10 | `OutTL_En` OR `Feature.Bit23` |
| 11 | `OutWL_En` OR `Feature.Bit23` |
| 12 | `OutAL_En` OR `Feature.Bit23` |
| 13 | `OutAH_MsgEn` |
| 14 | `OutWH_MsgEn` |
| 15 | `OutTH_MsgEn` |
| 16 | `OutTL_MsgEn` |
| 17 | `OutWL_MsgEn` |
| 18 | `OutAL_MsgEn` |
| 19 | Not used |
| 20 | Count up |

| Status bit | Parameter |
|---|---|
| 21 | Counter off |
| 22 | Count down |
| 23 - 28 | Not used |
| 29 | 1= Suppressed messages |
| 30 | Not used |
| 31 | MS_RelOp |

### See also

CountScL functions (Page 2080)

CountScL messaging (Page 2084)

CountScL block diagram (Page 2090)

CountScL error handling (Page 2084)

CountScL modes (Page 2079)

## 11.1.2 CountScL modes

### CountScL operating modes

This block provides the following operating modes:

- On (Page 77)
- Out of service (Page 77)

### "On"

General information on the "On" mode is available in the section On (Page 77).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit4` =1 and `Feature.Bit4` =0 is as shown below:

| Output Parameters | **Feature.Bit4 =0** | **Feature.Bit4 =1** |
|---|---|---|
| OutAH_Act.ST | 16#XX | 16#60 |
| OutWH_Act.ST | 16#XX | 16#60 |
| OutTH_Act.ST | 16#XX | 16#60 |
| OutTL_Act.ST | 16#XX | 16#60 |
| OutWL_Act.ST | 16#XX | 16#60 |
| OutAL_Act.ST | 16#XX | 16#60 |

**See also**

CountScL block diagram (Page 2090)
CountScL I/Os (Page 2085)
CountScL messaging (Page 2084)
CountScL error handling (Page 2084)
Description of CountScL (Page 2075)
CountScL functions (Page 2080)

## 11.1.3 CountScL functions

### Functions of CountScL

The functions for this block are listed below.

### Limit monitoring of the count value

This block provides the standard function Limit monitoring of the count value (Page 96).

### Group display SumMsgAct for limit monitoring, CSF and ExtMsgx

The block provides the standard function Group display for limit monitoring, CSF and ExtMsgx (Page 92).

### Read back the last counted value

When counting (visible at the `Out` output parameter), this count value is passed directly to the `OldOut` input parameter:

`OldOut` = `Out` If a warm restart is performed at this point, the count value (`Out`) is automatically reset to the default value, if the `Feature` Bit Set startup characteristics (Page 148) is set accordingly (= 0) (`OldOut` ≠ `Out`). In this case, the value from `OldOut` is only updated when the (`Out`) count value is changed again by a pulse. `OldOut` = `Out` now applies again.

### Reset counter to zero

The counted value at the `Out` output parameter is reset with the interconnectable `ResetCount` parameter. The reset is made with a 0 - 1 edge.

The count cannot be reset to zero from the faceplate.

You can use `Feature.Bit30` "Set reset depending on operating mode or the `LiOp` parameter" to configure a reset depending on the `LiOp` parameter:

`Feature.Bit30` = 0: The reset does not depend on `LiOp`

`Feature.Bit30` = 1: The reset is only possible when `LiOp =1`.

### Setting the count value to the default setting

You can use the PresetVal input parameter to enter a value at which the count should start, if the command has been given for setting the count to the default via the PresetEn input parameter with 1. This can also be done with the faceplate.

In this case, the Out output parameter is set to the PresetVal value.

You can use Feature.Bit30 "Set reset depending on operating mode or the LiOp parameter" to configure the setting of the count value to a default depending on the LiOp parameter:

Feature.Bit30 = 0: Setting the count value to the default setting does not depend on LiOp

Feature.Bit30 = 1: Setting the count value to the default is only possible if LiOp =0.

### Suppressing messages using the MsgLock parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status ST_Worst for the block is formed from the following parameter:

- In.ST
- OutST

The following signal status is formed by:

| Signal status | Used status |
|---|---|
| OutST | Count up, count down: Formed from the signal status In.ST and the last signal status OutST.<br>Counter off: Signal status OutST is frozen.<br>Resetting CountScL also resets the signal status: OutST := 16#80 |

### Configurable reactions using the Feature parameter

You can find an overview of all reactions provided by the Feature I/O in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |

| Bit | Function |
|---|---|
| 3 | Disable limit act output when limit value message is disabled (Page 209) |
| 4 | Changes signal status of outputs in OOS (Page 208) |
| 5 | Use the last value following a complete download as the current value during startup of the block (Page 165) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 23 | Limits disabled in faceplate (Page 206) |
| 24 | Enabling local operator authorization (Page 169) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 28 | Disabling operating points (Page 156) |
| 29 | Signaling limit violation (Page 181) |

## Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can stop counting |
| 5 | 1 = Operator can switch to incrementing mode |
| 6 | 1 = Operator can switch to decrementing mode |
| 7 - 11 | Not used |
| 12 | 1 = Operator can activate the Release for maintenance function |
| 13 | 1 = Operator can change the limit (`OutAH_Lim`) for high alarm |
| 14 | 1 = Operator can change the limit (`OutWH_Lim`) for high warning |
| 15 | 1 = Operator can change the limit (`OutTH_Lim`) for high tolerance |
| 16 | Not used |
| 17 | 1 = Operator can change the limit (`OutAL_Lim`) for low alarm |
| 18 | 1 = Operator can change the limit (`OutWL_Lim`) for low warning |
| 19 | 1 = Operator can change the limit (`OutTL_Lim`) for low tolerance |
| 20 | Not used |
| 21 | 1 = Operator can set the count to the default value (`PresetEn`) |
| 22 | 1 = Operator can specify the default value (`PresetTime`) |
| 23 | 1 = Operator can activate / deactivate messages via `OutAH_MsgEn` |
| 24 | 1 = Operator can activate / deactivate messages via `OutWH_MsgEn` |
| 25 | 1 = Operator can activate / deactivate messages via `OutTH_MsgEn` |
| 26 | 1 = Operator can activate / deactivate messages via `OutTL_MsgEn` |
| 27 | 1 = Operator can activate / deactivate messages via `OutWL_MsgEn` |
| 28 | 1 = Operator can activate / deactivate messages via `OutAL_MsgEn` |
| 29 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

### Providing limit at the output

For further connections to other blocks the following input parameters will also be displayed on the output:

`OutAH_LimOut` := `OutAH_Lim`

`OutWH_LimOut` := `OutWH_Lim`

`OutTH_LimOut` := `OutTH_Lim`

`OutTL_LimOut` := `OutTL_Lim`

`OutWL_LimOut` := `OutWL_Lim`

`OutAL_LimOut` := `OutAL_Lim`

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of CountScL (Page 2075)

CountScL messaging (Page 2084)

CountScL I/Os (Page 2085)

CountScL block diagram (Page 2090)

CountScL error handling (Page 2084)

CountScL modes (Page 2079)

## 11.1.4 CountScL error handling

### Error handling of CountScL

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 51 | Invalid signal at `LiOp` = 1:<br>• `OffLi` = 1 and `UpLi` = 1 and/or `DnLi` = 1<br>• `OffLi` = 0 and `UpLi` = 1 and `DnLi` = 1 |

### See also

CountScL block diagram (Page 2090)

CountScL I/Os (Page 2085)

CountScL messaging (Page 2084)

Description of CountScL (Page 2075)

CountScL modes (Page 2079)

CountScL functions (Page 2080)

## 11.1.5 CountScL messaging

### Messaging

The following messages can be generated for this block:

- Functions for displaying measured limits

Messages generated as a reaction to limit violations, can be suppressed by the settings `MsgEn` and `MsgLock`.

### Associated values for message instance `MsgEvId`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId` | SIG 1 | Alarm - high | $$BlockComment$$<br>High alarm limit violated |
| | SIG 2 | Warning - high | $$BlockComment$$<br>High warning limit violated |
| | SIG 3 | Tolerance - high | $$BlockComment$$<br>High tolerance limit violated |
| | SIG 4 | Tolerance - low | $$BlockComment$$<br>Low tolerance limit violated |
| | SIG 5 | Warning - low | $$BlockComment$$<br>Low warning limit violated |
| | SIG 6 | Alarm - low | $$BlockComment$$<br>Low alarm limit violated |
| | SIG 7 | Reserved | $$BlockComment$$ |
| | SIG 8 | Reserved | $$BlockComment$$ |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### See also

Description of CountScL (Page 2075)

CountScL functions (Page 2080)

CountScL I/Os (Page 2085)

CountScL block diagram (Page 2090)

CountScL modes (Page 2079)

CountScL error handling (Page 2084)

## 11.1.6 CountScL I/Os

I/Os of CountScL

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

## Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| DnLi | 1 = Down counter, via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DnOp* | 1 = Down counter, via operator | BOOL | 0 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 2080) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| In | Binary input value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| LiOp | 1 = Interconnection<br>0 = Operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgEvId | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_RelOp* | Operator can switch release for maintenance | BOOL | 0 |
| Occupied | 1 = In use by a batch | BOOL | 0 |
| OffLi | 1 = Counter disabled via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OffOp* | 1 = Counter disabled via operator | BOOL | 1 |
| OldOut* | Previous output value | DINT | 0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator permissions (Page 2080) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| OutAH_En | 1 = Enable alarm (high) for count | BOOL | 1 |
| OutAH_Lim | Limit for count alarm (high) | DINT | 95 |
| OutAH_MsgEn | 1 = Enable message for count alarm (high) | BOOL | 1 |
| OutAL_En | 1 = Enable alarm (low) for count | BOOL | 1 |
| OutAL_Lim | Limit for count alarm (low) | DINT | 0 |
| OutAL_MsgEn | 1 = Enable message for count alarm (low) | BOOL | 1 |
| OutOpHiScale | High limit for scale in count bar graph of faceplate | DINT | 100 |
| OutOpLoScale | Low limit for scale in count bar graph of faceplate | DINT | 0 |
| OutTH_En | 1 = Enable tolerance message (high) | BOOL | 0 |
| OutTH_Lim | Count tolerance limit message (high) | DINT | 85 |
| OutTH_MsgEn | 1 = Enable message for count tolerance message (high) | BOOL | 1 |
| OutTL_En | 1 = Enable count tolerance message (low) | BOOL | 0 |
| OutTL_Lim | Count tolerance message limit (low) | DINT | 0 |
| OutTL_MsgEn | 1 = Enable message for count tolerance message (low) | BOOL | 1 |
| OutUnit | Unit of measure for count Out | INT | 0 |
| OutWH_En | 1 = Enable count warning (high) | BOOL | 1 |
| OutWH_Lim | Count warning limit (high) | DINT | 90 |
| OutWH_MsgEn | 1 = Enable message for count warning (high) | BOOL | 1 |
| OutWL_En | 1 = Enable count warning (low) | BOOL | 1 |
| OutWL_Lim | Count warning limit (low) | DINT | 0 |
| OutWL_MsgEn | 1 = Enable message for count warning (low) | BOOL | 1 |
| PresetEn* | 1 = Set block to default count (PresetVal) | BOOL | 0 |
| PresetVal | Default setting for the count | DINT | 0 |
| ResetCount | 1 = Counter restart | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RunUpCyc | Number of operating cycles for which all messages are suppressed | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SelFp2 | Call a block saved in this parameter as an additional face-plate (Page 234) in the preview | ANY | |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UpLi | 1 = Forward counter, via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| UpOp* | 1 = Forward counter, via operator | BOOL | 0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| CountMode | 0 = Counter off<br>1 = Count up<br>2 = Count down | INT | 0 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ER_AL_Out | ER alarm high limit Output | DINT | 0 |
| ER_AH_Out | ER alarm low limit Output | DINT | 0 |
| ER_TH_Out | ER Tolerance High Limit Output | DINT | 0 |
| ER_TL_Out | ER Tolerance Low Limit Output | DINT | 0 |
| ER_WH_Out | ER Warning High Limit Output | DINT | 0 |
| ER_WL_Out | ER Warning Low Limit Output | DINT | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see CountScL error handling (Page 2084) | INT | -1 |
| MsgAckn | Message acknowledgement status (output ACK_STATE of first ALARM_8P) | WORD | 16#0000 |
| MsgErr | 1 = Message error (output ERROR of the first ALARM_8P) | BOOL | 0 |
| MsgStat | Message status (output STATUS of the first ALARM_8P) | WORD | 16#0000 |
| MS_Release | Release for maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| O_MS_Ext | Reserved | DWORD | 0 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| Out | Count | DINT | 0 |
| OutAH_Act | 1 = Count alarm (high) enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OutAL_Act | 1 = Count alarm (low) enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OutOpHiScaleOut | OutOpHiScale as output | DINT | 0 |
| OutOpLoScaleOut | OutOpLiScale as output | DINT | 0 |
| OutST | Signal status for Out | BYTE | 16#80 |
| OutTH_Act | 1 = Count tolerance message (high) enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OutTL_Act | 1 = Count tolerance message (low) enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OutUnitOut | OutUnit as output | INT | 0 |
| OutWH_Act | 1 = Count warning (high) enabled. You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OutWL_Act | 1 = Count warning (low) enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OutAH_LimOut | OutAH_Lim as output | DINT | 0 |
| OutWH_LimOut | OutWH_Lim as output | DINT | 0 |
| OutTH_LimOut | OutTH_Lim as output | DINT | 0 |
| OutAL_LimOut | OutAL_Lim as output | DINT | 0 |
| OutWL_LimOut | OutWL_Lim as output | DINT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| `OutTL_LimOut` | `OutTL_Lim` as output | DINT | 0 |
| `ST_Worst` | Worst signal status | BYTE | 16#80 |
| `Status1` | Status word 1 (Page 2075) | DWORD | 16#00000000 |
| `Status2` | Status word 2 (Page 2075) | DWORD | 16#00000000 |

**See also**

CountScL messaging (Page 2084)

CountScL block diagram (Page 2090)

CountScL modes (Page 2079)

## 11.1.7 CountScL block diagram

### CountScL block diagram

A block diagram is not provided for this block.

**See also**

Description of CountScL (Page 2075)

CountScL modes (Page 2079)

CountScL functions (Page 2080)

CountScL error handling (Page 2084)

CountScL messaging (Page 2084)

CountScL I/Os (Page 2085)

## 11.1.8 Operator control and monitoring

### 11.1.8.1 CountScL views

### Views of the CountScL block

The block CountScL provides the following views:

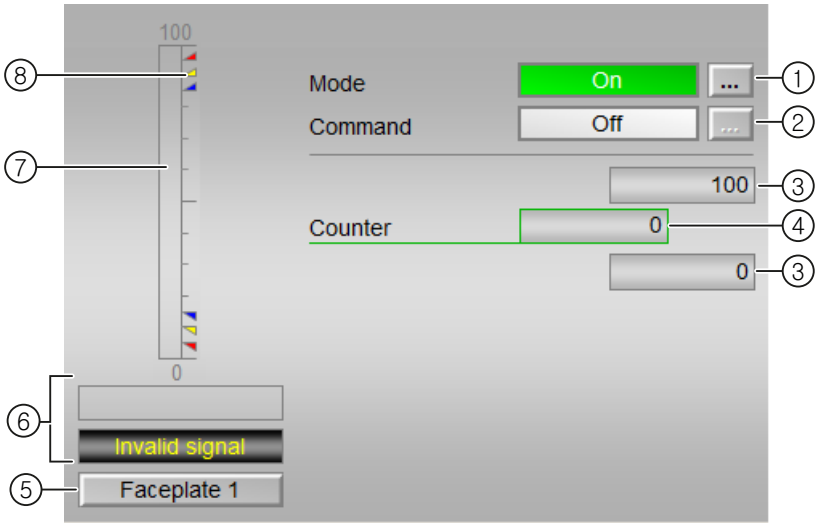
- CountScL standard view (Page 2091)
- Alarm view (Page 336)
- CountScL limit view (Page 2093)
- Trend view (Page 341)

- CountScL parameter view (Page 2094)
- CountScL preview (Page 2095)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for CountScL (Page 2096)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 11.1.8.2 CountScL standard view

#### CountScL standard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 77)
- Out of service (Page 77)

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the operating mode.

### (2) Enabling and disabling the counter

This area shows you the default operating state for the counter. The following states can be shown and executed here:

- "On ⇑"
- "On ⇓"
- "Off"

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

### (3) High and low scale range for the count value

These values provide information on the display range for the bar graph of the count. The scale range is defined in the engineering system.

### (4) Display of the count value

This area shows the current count value with the signal status `OutST`.

The name for the current count value can be set using the `s7_shortcut` attribute at the corresponding output parameter. The default text is displayed if nothing is specified.

You can find additional information on this in the section Labeling of buttons and text (Page 235) .

### (5) Navigation button for switching to the standard view of any faceplate

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234) .

### (6) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"
- "Internal error"
- "Invalid Signal"

### (7) Graphic display of the current count value

This area shows you the current count in form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (8) Limit display

These colored triangles indicate the configured limits in the respective bar graph:

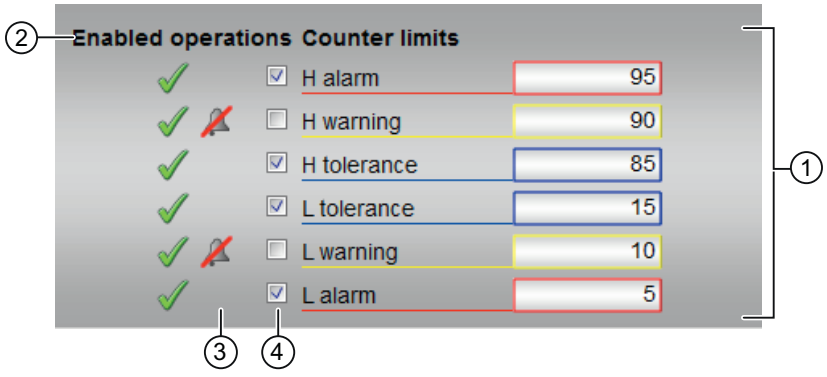
- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

In counting mode "count up", only the colored triangles of the upper limits are visible; and in counting mode "count down", only the colored triangles of the lower limits are visible.

For more information, refer: Bar graph (Page 376)

## 11.1.8.3 CountScL limit view

### Limit view of CountScL




### (1) Limits for the counter

In this area, you can enter the limits for the counter. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H alarm": Alarm high
- "H warning": Warning high
- "H tolerance": Tolerance high
- "L tolerance": Tolerance low
- "L warning": Warning low
- "L alarm": Alarm low

### (2) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

### (3) "Message suppression"

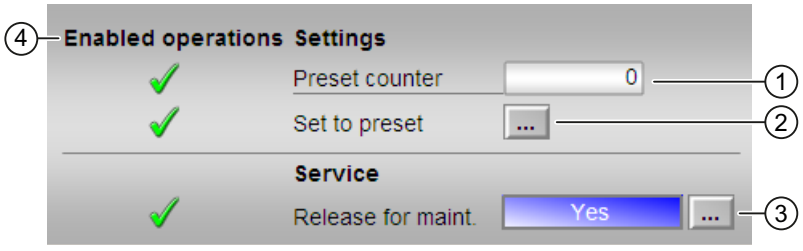
Message suppression indicates whether or not the suppression of the associated message in the AS block is activated with the `xx_MsgEn` parameter. The output of messages is not suppressed when the block is installed (all `xx_MsgEn` parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

### (4) Suppress messages

You can enable / disable messages by setting the check mark.

## 11.1.8.4 CountScL parameter view

### Parameter view of CountScL




### (1) Preset counter

Enter the default setting here, where the counter should start. You can find additional information on this in the Changing values (Page 291) section.

### (2) Set to default

Set the counter to the default value here. You can find additional information on this in the Switching operating states and operating modes (Page 289) section.

### (3) Service

You can select the following function in this area:

- "Release for maintenance"

Refer to the Switching operating states and operating modes (Page 289) section for more on this.

You can find information on this area in the Release for maintenance (Page 70) section.

### (4) Enabled operations

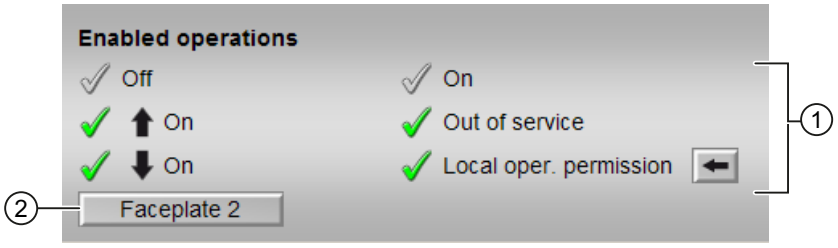
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

## 11.1.8.5 CountScL preview

### Preview of CountScL




### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "Off": You can disable the counter.
- "On ⇑": You can operate the incremental counter.
- "On ⇓": You can operate the decremental counter.

- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. Additional information is available in the section Operator control permissions (Page 285).

### (2) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Opening additional faceplates (Page 234) section.

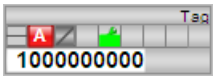
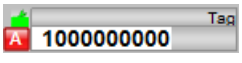
## 11.1.8.6 Block icon for CountScL

### Block icons for CountScL

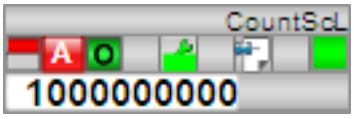
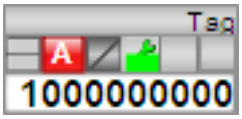
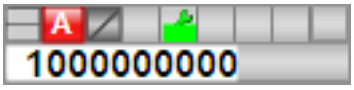
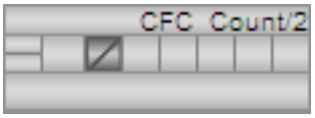
A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits
- Operating modes
- Signal status, release for maintenance
- Memo display
- Display counter running

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in the CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | Block icon in the full display |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269).

# 11.2 CountOh - Determining runtime

## 11.2.1 Description of CountOh

### Object name (type + number) and family

Type + number: FB 1864

Family: Count

### Area of application for CountOh

The block is used for the following applications:

- Calculating the runtime of a unit

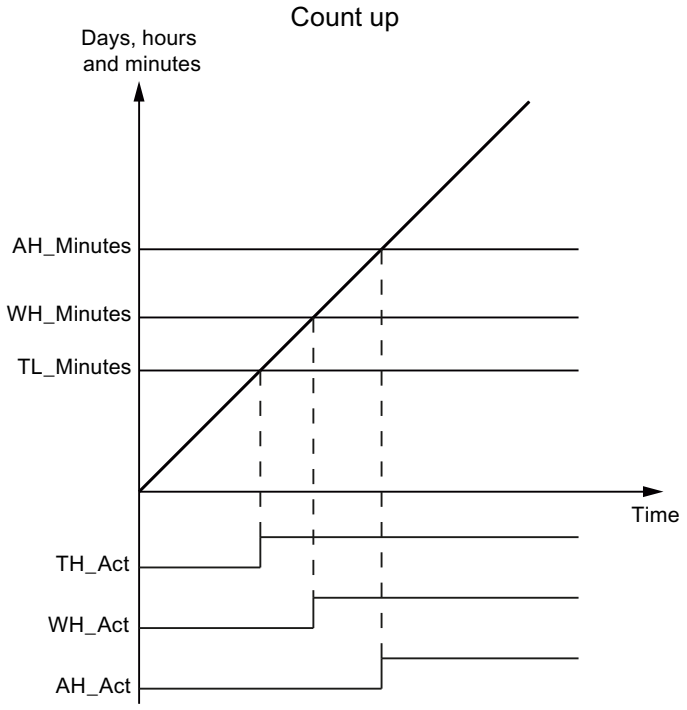
### How it works

The block calculates the operating time of a unit.

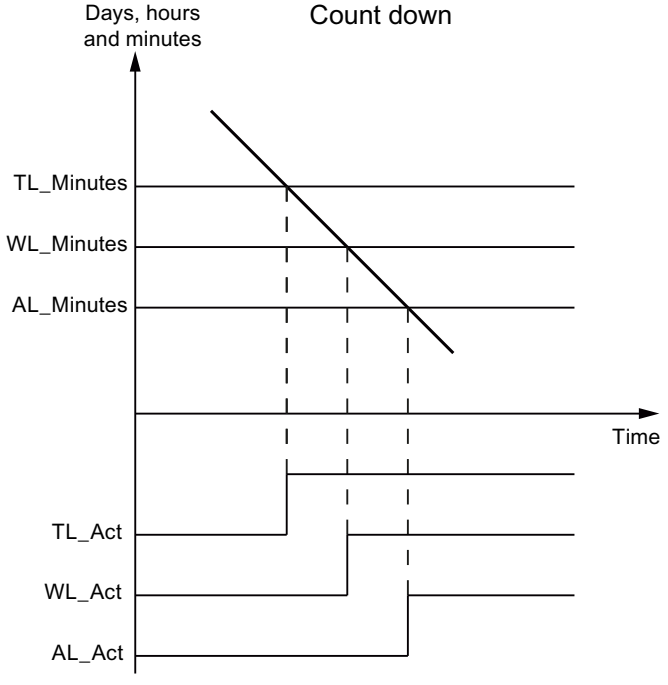
1. Off (`OffOp` = 1 or `OffLi` = 1)
   The block is disabled (output parameter `CountMode` = 0). No counting takes place.
2. Count up (`UpOp` = 1 or `UpLi` = 1)
   The operating time of the connected unit is incremented (output parameter `CountMode` = 1).
3. Count down (`DnOp` = 1 or `DnLi` = 1)
   The operating time of the connected unit is decremented (output parameter `CountMode` = 2).

The operating time is shown depending on the course of days, hours, minutes and seconds.

If the operating time exceeds the configured limits, an alarm is triggered.




When the block counts up and the total operating time `(TimeMin)` of the connected unit is greater than or equal to the limits (`TH_Minutes`, `WH_Minutes` and `AH_Minutes`), the alarms `TH_Act`, `WH_Act` and `AH_Act` are set.




When the block counts up and the total operating time `(TimeMin)` of the connected unit is greater than or equal to the limits (`TL_Minutes`, `WL_Minutes` and `AL_Minutes`), the alarms `TL_Act`, `WL_Act` and `AL_Act` are set.

The operating time can be preset. The value can be specified by the operator. The operator can set values for days, hours and minutes to begin counting, if the required operator control permission has been issues for this.

The maximum configurable value for the operating time is 24855 days, 3 hours and 14 minutes.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is first installed automatically in the startup OB (OB 100).

### Startup characteristics

Define the startup characteristics for this block via two `Feature` bits:

Bit 0: Set startup characteristics (Page 148)

Bit 5: Use the last value following a complete download as the current value during startup of the block (Page 165)

When `Feature.Bit5` is set to 1, then:

- `Days := OldDays`
- `Hours := OldHours`
- `Minutes := OldMinutes`
- `Seconds := OldSeconds`

When you set `Feature.Bit5` and `Feature.Bit0` to 1, then:

- `TotalTime := PresetTime`

The output parameters `Days`, `Hours`, `Minutes` are converted accordingly based on `PresetTime`.

When you set `Feature.Bit5` to 0 and `Feature.Bit0` to 1, no preset is made.

### Time response

The block only functions properly in a cyclic interrupt OB. To ensure correct time acquisition, it should be installed (if configured in CFC) in the same runtime group as the control block of the monitored unit.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section CountOh I/Os (Page 2109).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | Not used |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |

| Status bit | Parameter |
|---|---|
| 5 | Not used |
| 6 | OnAct.Value |
| 7 - 10 | Not used |
| 11 | LiOp |
| 12 - 13 | Not used |
| 14 | 1 = Invalid signal status |
| 15 | Not used |
| 16 | Internal error |
| 17-31 | Not used |

### Status word allocation for Status2 parameter

| Status bit | Parameter |
|---|---|
| 0 | MsgLock.Value |
| 1 | AH Act.Value |
| 2 | WH_Act.Value |
| 3 | TH_Act.Value |
| 4 | TL_Act.Value |
| 5 | WL_Act.Value |
| 6 | AL_Act.Value |
| 7 | AH_En OR Feature.Bit23 |
| 8 | WH_En OR Feature.Bit23 |
| 9 | TH_En OR Feature.Bit23 |
| 10 | TL_En OR Feature.Bit23 |
| 11 | WL_En OR Feature.Bit23 |
| 12 | AL_En OR Feature.Bit23 |
| 13 | AH_MsgEn |
| 14 | WH_MsgEn |
| 15 | TH_MsgEn |
| 16 | TL_MsgEn |
| 17 | WL_MsgEn |
| 18 | AL_MsgEn |
| 19 | Not used |
| 20 | Count up |
| 21 | Counter off |
| 22 | Count down |
| 23 - 28 | Not used |
| 29 | 1= Suppressed messages |
| 30 | Not used |
| 31 | MS_RelOp |

### See also

CountOh functions (Page 2103)

CountOh messaging (Page 2108)

CountOh block diagram (Page 2114)

CountOh error handling (Page 2107)

CountOh modes (Page 2102)

## 11.2.2 CountOh modes

### CountOh operating modes

The block can be operated using the following modes:

- On (Page 77)
- Out of service (Page 77)

### "On"

You can find general information about the "On" mode in the On (Page 77) section.

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit4` =1 and `Feature.Bit4` =0 is as shown below:

| Output Parameters | **`Feature.Bit4` =0** | **`Feature.Bit4` =1** |
|---|---|---|
| DeviceOn.ST | 16#XX | 16#60 |
| AH_Act.ST | 16#XX | 16#60 |
| WH_Act.ST | 16#XX | 16#60 |
| TH_Act.ST | 16#XX | 16#60 |
| TL_Act.ST | 16#XX | 16#60 |
| WL_Act.ST | 16#XX | 16#60 |
| AL_Act.ST | 16#XX | 16#60 |

### See also

CountOh block diagram (Page 2114)

CountOh I/Os (Page 2109)

CountOh messaging (Page 2108)

CountOh error handling (Page 2107)

Description of CountOh (Page 2098)
CountOh functions (Page 2103)

## 11.2.3 CountOh functions

### Functions of CountOh

The block provides the following functions:

### Limit monitoring of the operating time

The total time in minutes `(TimeIn)` is displayed for the following limits:

1. Alarm (`AH_Minutes` and `AL_Minutes`)
2. Warning (`WH_Minutes` and `WL_Minutes`)
3. Tolerance (`TH_Minutes` and `TL_Minutes`)

**Note**

**Limit monitoring:**

Limit monitoring depends on the counting direction:

- Operation mode 1 (counting up), limit monitoring top
  - `AH_Minutes`
  - `WH_Minutes`
  - `TH_Minutes`
- Operation mode 2 (counting down), limit monitoring bottom
  - `AL_Minutes`
  - `WL_Minutes`
  - `TL_Minutes`

### Group display SumMsgAct for limit monitoring, CSF and ExtMsgx

The block provides the standard function Group display for limit monitoring, CSF and ExtMsgx (Page 92).

### Suppressing messages using the `MsgLock` parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 3 | Disable limit act output when limit value message is disabled (Page 209) |
| 4 | Changes signal status of outputs in OOS (Page 208) |
| 5 | Use the last value following a complete download as the current value during startup of the block (Page 165) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 23 | Limits disabled in faceplate (Page 206) |
| 24 | Enabling local operator authorization (Page 169) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 28 | Disabling operating points (Page 156) |
| 29 | Signaling limit violation (Page 181) |

### Read back the last counted value

When counting (`DeviceOn.Value = 1` and `CountMode = 1` or `2`), the inputs of the most recently valid operating time are updated:

- `OldDays := Days`
- `OldHours := Hours`
- `OldMinutes := Minutes`
- `OldSeconds := Seconds`

The inputs `OldDays`, `OldHours`, `OldMinutes`, `OldSeconds` are read back with a complete download.

After a warm restart, the outputs of the operating time are set depending on the configuration of the `Feature` parameter.

When `Feature.Bit5` is set to 1, then:

- `Days := OldDays`
- `Hours := OldHours`
- `Minutes := OldMinutes`
- `Seconds := OldSeconds`

When you set `Feature.Bit5` and `Feature.Bit0` to 1, then:

- `TotalTime := PresetTime`

The output parameters `Days`, `Hours`, `Minutes` are converted accordingly based on `PresetTime`.

When you set `Feature.Bit5` to 0 and `Feature.Bit0` to 1, no preset is made.

### Display and operator input area for process values and setpoints

This block provides the standard function Display and operator input area for process values and setpoints (Page 233).

### Reset counter to zero

The output parameters of the operating time are reset via the interconnectable `Reset` parameter. The reset is made with a 0 - 1 edge.

The count cannot be reset to zero from the faceplate.

You can use `Feature.Bit30` "Set reset depending on operating mode or the LiOp parameter" to configure a reset depending on the LiOp parameter:

`Feature.Bit30` = 0: The reset does not depend on `LiOp`

`Feature.Bit30` = 1: The reset is only possible when `LiOp =1`

### Setting the count value to the default setting

You can enter the operating time at which counting should begin with the input parameters `PresetTime` and `PresetEn`.

When `PresetEn` is set, the following applies:

- `TotalTime := PresetTime`

The outputs `Days`, `Hours`, `Minutes` are converted accordingly based on `PresetTime`. `PresetTime` can also be entered in the faceplate.

The `PresetTime` time value is always indicated in seconds.

You can use the `Feature.Bit30` "Set reset depending on operating mode or the `LiOp` parameter" to configure the setting of the count value to a default depending on the `LiOp` parameter:

`Feature.Bit30` = 0: Setting to the default does not depend on `LiOp`

`Feature.Bit30` = 1: Setting to the default is only possible if `LiOp` =0.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameter:

- `In.ST`
- `TmMinSt`

The following signal status is formed by:

| Signal status | Used status |
|---|---|
| TmMinST | Count up, count down: Formed from the signal status In.ST and the last signal status TmMinST.<br>Counter off: Signal status TmMinST is frozen.<br>Resetting CountOh also resets the signal status: TmMinST := 16#80 |

## Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the OS_Perm parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can start the block |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can stop the block |
| 5 | 1 = Operator can switch to incrementing mode |
| 6 | 1 = Operator can switch to decrementing mode |
| 7 - 11 | Not used |
| 12 | 1 = Operator can enable release for maintenance |
| 13 | 1 = Operator can enter AH_Lim |
| 14 | 1 = Operator can enter WH_Lim |
| 15 | 1 = Operator can enter TH_Lim |
| 16 | Not used |
| 17 | 1 = Operator can enter AL_Lim |
| 18 | 1 = Operator can enter WL_Lim |
| 19 | 1 = Operator can enter TL_Lim |
| 20 | Not used |
| 21 | 1 = Operator can enable default |
| 22 | 1 = Operator can apply configured time |
| 23 | 1 = Operator can activate / deactivate messages via AH_MsgEn |
| 24 | 1 = Operator can activate / deactivate messages via WH_MsgEn |
| 25 | 1 = Operator can activate / deactivate messages via TH_MsgEn |
| 26 | 1 = Operator can activate / deactivate messages via TL_MsgEn |
| 27 | 1 = Operator can activate / deactivate messages via WL_MsgEn |
| 28 | 1 = Operator can activate / deactivate messages via AL_MsgEn |
| 29 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in OS_Perm as a parameter, you have to reset the corresponding OS_Perm bit.

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of CountOh (Page 2098)

CountOh messaging (Page 2108)

CountOh I/Os (Page 2109)

CountOh block diagram (Page 2114)

CountOh error handling (Page 2107)

CountOh modes (Page 2102)

## 11.2.4 CountOh error handling

### Error handling of CountOh

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 51 | Invalid signal at `LiOp` = 1<br>• `OffLi` = 1 and `UpLi` = 1 and/or `DnLi` = 1<br>• `OffLi` = 0 and `UpLi` = 1 and `DnLi` = 1 |

### See also

CountOh block diagram (Page 2114)

CountOh I/Os (Page 2109)

CountOh messaging (Page 2108)

CountOh functions (Page 2103)
CountOh modes (Page 2102)
Description of CountOh (Page 2098)

## 11.2.5 CountOh messaging

### Messaging

The following messages can be generated for this block:

- Functions for displaying measured limits

Messages generated as a reaction to limit violations, can be suppressed by the settings `xx_MsgEn` and `MsgLock`.

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId 1` | SIG 1 | Alarm - high | $$BlockComment$$<br>High alarm limit violated |
| | SIG 2 | Warning - high | $$BlockComment$$<br>High warning limit violated |
| | SIG 3 | Tolerance - high | $$BlockComment$$<br>High tolerance limit violated |
| | SIG 4 | Tolerance - low | $$BlockComment$$<br>Low tolerance limit violated |
| | SIG 5 | Warning - low | $$BlockComment$$<br>Low warning limit violated |
| | SIG 6 | Alarm - low | $$BlockComment$$<br>Low alarm limit violated |
| | SIG 7 | Reserved | $$BlockComment$$<br>For internal use |
| | SIG 8 | Reserved | $$BlockComment$$<br>For internal use |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### Associated values for message instance `MsgEvId1`

| Associated value | Block parameters |
|---|---|
| 1 | BatchName |
| 2 | StepNo |
| 3 | BatchID |

### See also

Description of CountOh (Page 2098)

CountOh functions (Page 2103)

CountOh I/Os (Page 2109)

CountOh block diagram (Page 2114)

CountOh error handling (Page 2107)

CountOh modes (Page 2102)

## 11.2.6 CountOh I/Os

I/Os of CountOh

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AH_En | High alarm enabled | BOOL | 1 |
| AH_MsgEn | Message for high alarm enabled | BOOL | 1 |
| AL_En | Low alarm enabled | BOOL | 1 |
| AL_MsgEn | Message for low alarm enabled | BOOL | 1 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch number (Batch ID) | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| BlkTypeInfo* | Block Type Version and FB Number of block-type<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| DayOpHiScale* | Operating stage - High limit of bar display for OS | INT | 100 |
| DayOpLoScale* | Operating stage - Low limit of bar display for OS | INT | 0 |
| DaysAHLim* | Days - high limit alarm | INT | 95 |
| DaysALLim* | Days - low limit alarm | INT | 0 |
| DaysTHLim* | Days - high limit tolerance | INT | 85 |
| DaysTLLim* | Days - low limit tolerance | INT | 0 |
| DaysWHLim* | Days - limit for high warning | INT | 90 |
| DaysWLLim* | Days - limit for low warning | INT | 0 |
| DnLi | 1 = Decrement via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DnOp* | 1 = Decrement via OS operator | BOOL | 0 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 2103) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| HrsAHLim* | Hours - high limit alarm | INT | 0 |
| HrsALLim* | Hours - low limit alarm | INT | 0 |
| HrsOpHiScale* | Operating hours - high limit of bar display for OS | INT | 23 |
| HrsOpLoScale* | Operating hours - low limit of bar display for OS | INT | 0 |
| HrsTHLim* | Hours - high limit tolerance | INT | 0 |
| HrsTLLim* | Hours - low limit tolerance | INT | 0 |
| HrsWHLim* | Hours - high limit warning | INT | 0 |
| HrsWLLim* | Hours - low limit warning | INT | 0 |
| In | Device status:<br>1=On<br>0=Off | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| LiOp | Switchover of operating mode between:<br>1 = Interconnection<br>0 = Operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MinOpHiScale* | Operating minutes - high limit of bar display for OS | INT | 59 |
| MinOpLoScale* | Operating minutes - low limit of bar display for OS | INT | 0 |
| MinsAHLim* | Minutes - high limit alarm | INT | 0 |
| MinsALLim* | Minutes - low limit alarm | INT | 0 |
| MinsTHLim* | Minutes - high limit tolerance | INT | 0 |
| MinsTLLim* | Minutes - low limit tolerance | INT | 0 |
| MinsWHLim* | Minutes - high limit warning | INT | 0 |
| MinsWLLim* | Minutes - low limit warning | INT | 0 |
| MS_RelOp* | Operator input for release for maintenance, 1: Release for maintenance demand | BOOL | 0 |
| MsgEvId | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1= Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = In use by a batch | BOOL | 0 |
| OffLi | Counter disabled via interconnection:<br>1 = Off | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OffOp* | Counter disabled via operator:<br>1 = Off | BOOL | 1 |
| OldDays* | Previous day value | INT | 0 |
| OldHours* | Previous hour value | INT | 0 |
| OldMinutes* | Previous minute value | INT | 0 |
| OldSeconds* | Previous second value | INT | 0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for CountOh functions (Page 2103) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PresetEn* | 1 = Set block to default time time value (PresetTime) | BOOL | 0 |
| PresetTime* | Default setting for the time value [s] | DWORD | 16#00000000 |
| Reset | 1 = Reset counter | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] | REAL | 0.1 |
| SelFp1 | Open faceplate 1 | ANY | |
| SelFp2 | Open faceplate 2 | ANY | |
| StepNo | Batch step number | DWORD | 16#00000000 |
| TH_En | High tolerance enabled | BOOL | 0 |
| TH_MsgEn | Alarm for high tolerance enabled | BOOL | 1 |
| TL_En | Low tolerance enabled | BOOL | 0 |
| TL_MsgEn | Alarm for low tolerance enabled | BOOL | 1 |
| UpLi | 1 = Increment (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| UpOp* | 1 = Increment (via faceplate) | BOOL | 0 |
| WH_En | High warning enabled | BOOL | 1 |
| WH_MsgEn | Alarm for high warning enabled | BOOL | 1 |
| WL_En | Low warning enabled | BOOL | 1 |
| WL_MsgEn | Alarm for low warning enabled | BOOL | 1 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AH_Act | High alarm enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| AH_Minutes | High alarm time [in min] | DINT | 0 |
| AL_Act | Low alarm enabled<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| AL_Minutes | Low alarm time [in min] | DINT | 0 |
| BarOpHiScale | High limit for bar display [min] | DINT | 100 |
| BarOpLoScale | Low limit for bar display [min] | DINT | 0 |
| CountMode | Count mode:<br>0 = Off<br>1 = Increment<br>2 = Count down | INT | 0 |
| Days | Operating stage | INT | 0 |
| DeviceOn | 1 = Unit on | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see CountOh error handling (Page 2107) | INT | -1 |
| Hours | Operating hours | INT | 0 |
| Minutes | Operating minutes | INT | 0 |
| MS_Release | Release for maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgAckn | Message acknowledgment status (output ACK_STATE of ALARM_8P) | WORD | 16#0000 |
| MsgErr | 1 = Message error (output ERROR of ALARM_8P) | BOOL | 0 |
| MsgStat | Message status (output STATUS of ALARM_8P) | WORD | 16#0000 |
| O_MS_Ext | Reserved | DWORD | 16#00000000 |
| OnAct | Block running | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | Block is "out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Operator permission: Output for OS | DWORD | 16#FFFFFFFF |
| OS_PermOut | Operator permission: Output for OS | DWORD | 16#FFFFFFFF |
| Seconds | Operating time [in sec] | INT | 0 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 2098) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 2098) | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| TH_Act | High tolerance enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| TH_Minutes | Tolerance high | DINT | 0 |
| TimeMin | Operating period [min] | DINT | 0 |
| TmMinST | Signal Station for TimeMin | BYTE | 16#80 |
| TL_Act | Low tolerance enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| TL_Minutes | Low tolerance time [in min] | DINT | 0 |
| TotalTime | Total operating time | DWORD | 16#00000000 |
| WH_Act | High warning enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| WH_Minutes | High warning time [in min] | DINT | 0 |
| WL_Act | Low warning enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| WL_Minutes | Low alarm time [in min] | DINT | 0 |

### See also

CountOh messaging (Page 2108)

CountOh block diagram (Page 2114)

CountOh modes (Page 2102)

## 11.2.7 CountOh block diagram

### CountOh block diagram

A block diagram is not provided for this block.

**See also**

CountOh I/Os (Page 2109)
CountOh messaging (Page 2108)
CountOh error handling (Page 2107)
CountOh functions (Page 2103)
CountOh modes (Page 2102)
Description of CountOh (Page 2098)

## 11.2.8 Operator control and monitoring

### 11.2.8.1 CountOh views

**Views of the CountOh block**

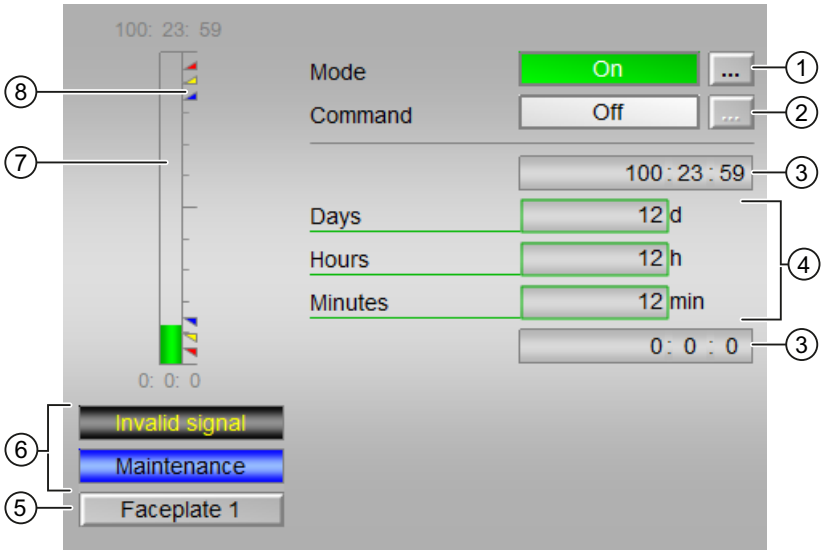
The block CountOh provides the following views:

- CountOh standard view (Page 2116)
- Alarm view (Page 336)
- CountOh limit view (Page 2118)
- Trend view (Page 341)
- CountOh parameter view (Page 2119)
- CountOh preview (Page 2120)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for CountOh (Page 2121)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 11.2.8.2 CountOh standard view

#### CountOh standard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 77)
- Out of service (Page 77)

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the operating mode.

#### (2) Enabling and disabling the counter

This area shows you the default operating state for the counter. The following states can be shown and executed here:

- "On ⇑"
- "On ⇓"
- "Off"

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

#### (3) High and low scale range for the count value

These values provide information on the display range for the bar graph **(5)** of the count value. The scale range is defined in the engineering system.

### (4) Displaying the counts values

The following counts are shown here:

- "Days" with signal status `TmMinST`
- "Hours" with signal status `TmMinST`
- "Minutes" with signal status `TmMinST`

### (5) Navigation button for switching to the standard view of any faceplate

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234) .

### (6) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"
- "Internal error"
- "Invalid Signal"

### (7) Graphic display of the current count value

This area shows you the current count in form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

### (8) Limits

These colored triangles indicate the configured limits in the respective bar graph:

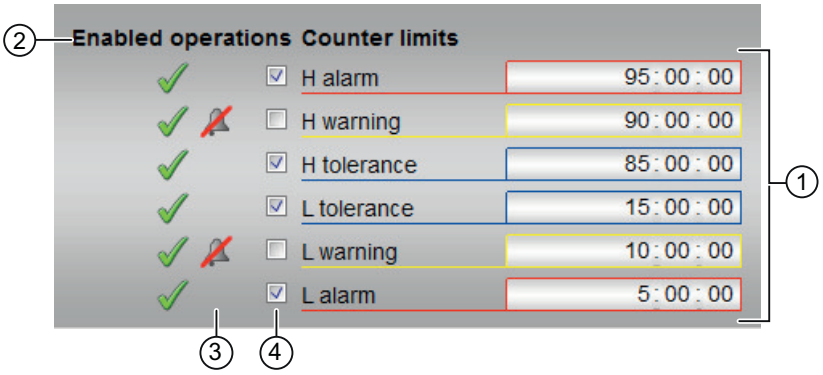
- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

In counting mode "count up", only the colored triangles of the upper limits are visible; and in counting mode "count down", only the colored triangles of the lower limits are visible.

For more information, refer: Bar graph (Page 376)

### 11.2.8.3 CountOh limit view

#### Limit view of CountOh




#### (1) Limits for the counter

In this area, you can enter the limits for the counter. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H alarm": Alarm high
- "H warning": Warning high
- "H tolerance": Tolerance high
- "L tolerance": Tolerance low
- "L warning": Warning low
- "L alarm": Alarm low

#### (2) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

### (3) "Message suppression"

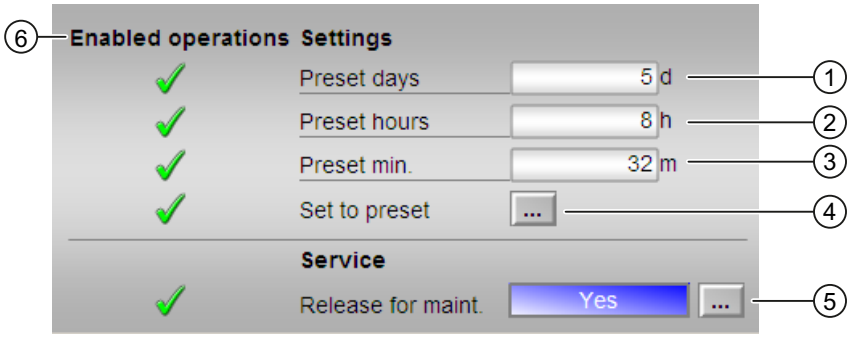
Message suppression indicates whether or not the suppression of the associated message in the AS block is activated with the `xx_MsgEn` parameter. The output of messages is not suppressed when the block is installed (all `xx_MsgEn` parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

### (4) Suppress messages

You can enable / disable messages by setting the check mark.

## 11.2.8.4 CountOh parameter view

### Parameter view of CountOh




### (1), (2) and (3) preset counter

Enter the default setting here, where the counter should start.

You can change the following presets:

- "Days"
- "Hours"
- "Minutes"

You can find additional information on this in the Changing values (Page 291) section.

### (4) Set to default

Set the counter to the default value here. You can find additional information on this in the Switching operating states and operating modes (Page 289) section.

### (5)Service

You can select the following function in this area:

- "Release for maintenance"

Refer to the Switching operating states and operating modes (Page 289) section for more on this.

You can find information on this area in the Release for maintenance (Page 70) section.

### (6) Enabled operations

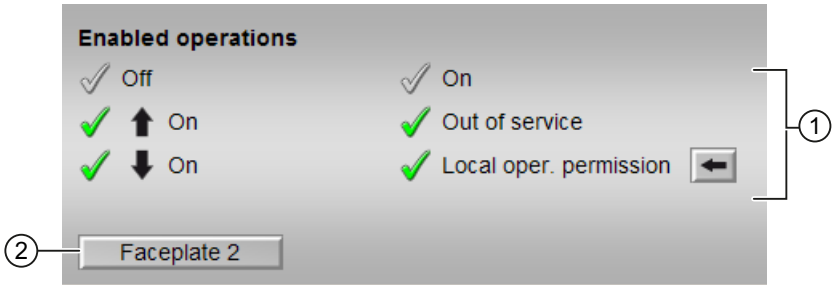
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

## 11.2.8.5 CountOh preview

### Preview of CountOh




### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "Off": You can disable the counter.
- "On ⇑": You can operate the incremental counter.

- "On ⇓": You can operate the decremental counter.
- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. Additional information is available in the section Operator control permissions (Page 285).

### (2) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Opening additional faceplates (Page 234) section.

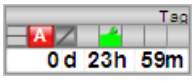
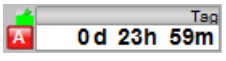
## 11.2.8.6 Block icon for CountOh

### Block icons for CountOh

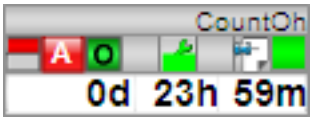
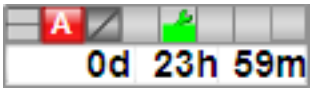
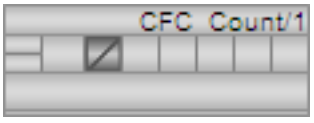
A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits
- Operating modes
- Signal status, release for maintenance
- Memo display
- Display counter running

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | Block icon in the full display |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269).

# 11.3 TotalL - Additive counter with upward or downward counting direction (totalizer)

## 11.3.1 Description of TotalL

### Object name (type + number) and family

Type + number: FB 1906

Family: Count

### Area of application for TotalL

The block is used for the following applications:

- Triggered upward or downward summing
- Continuous upward or downward summing
- Upward or downward integration of an analog input value

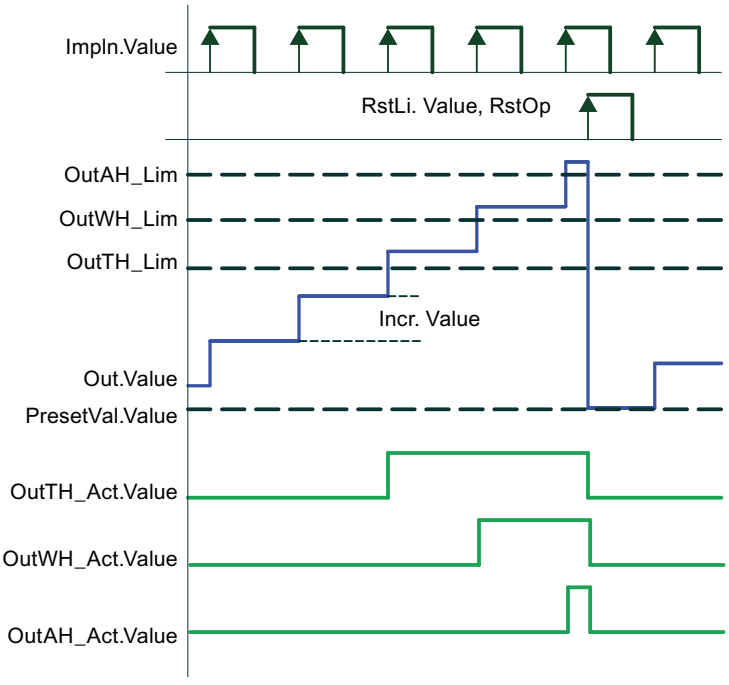
### How it works

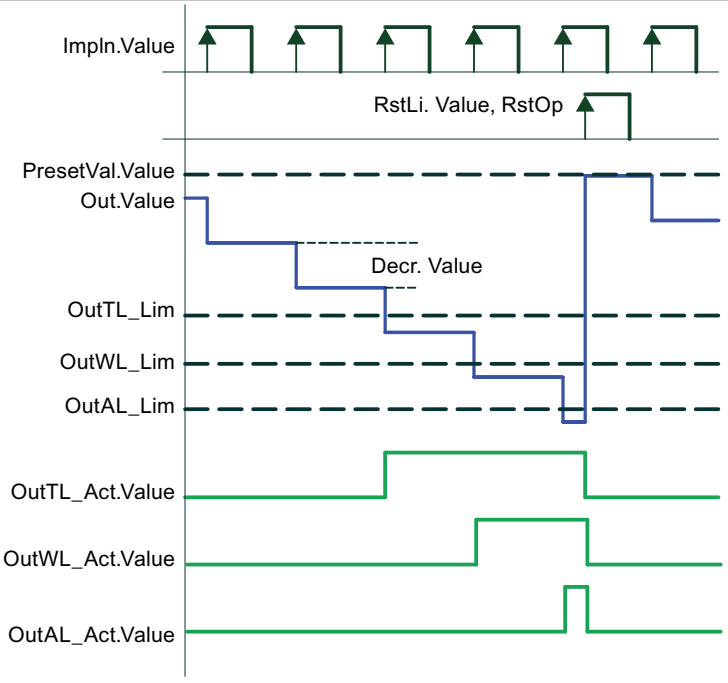
With a positive edge at the input parameter (pulse signal) `P_In` the output parameter `Out` is increased by a variable increment or reduced by a variable decrement.

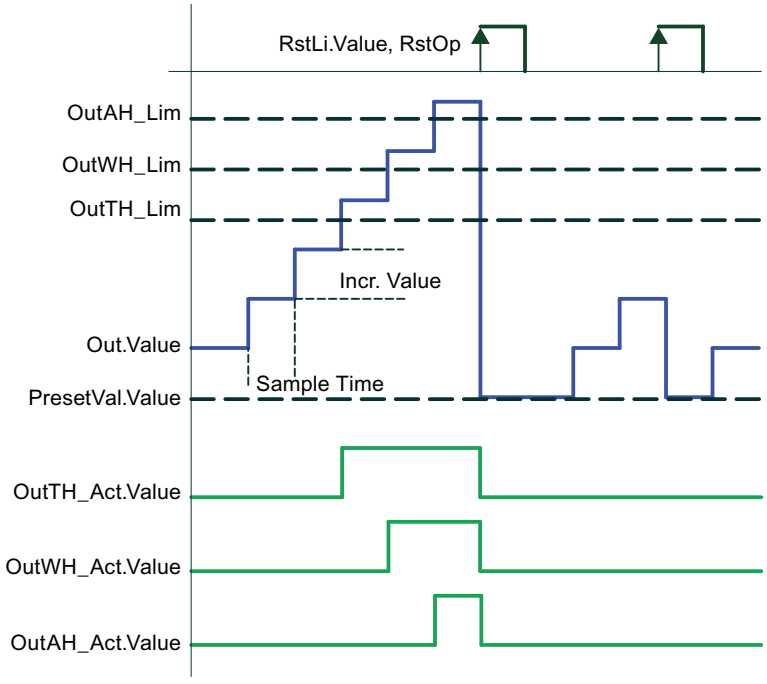
Configure how it works via the following `Feature` Bits:

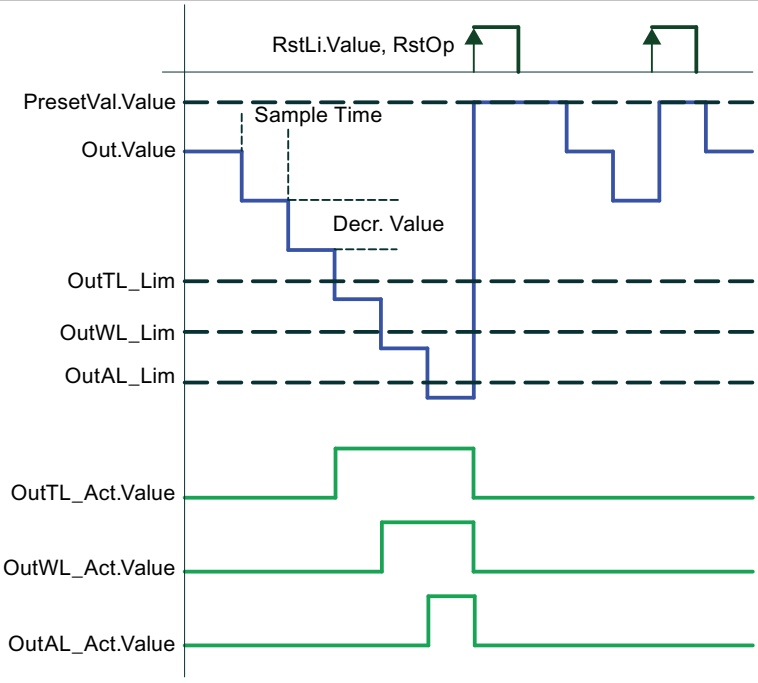
- `Feature.Bit6`: Block as summing unit or integrator (Page 154)
- `Feature.Bit7`: Summing characteristics continuous or triggered (Page 184)

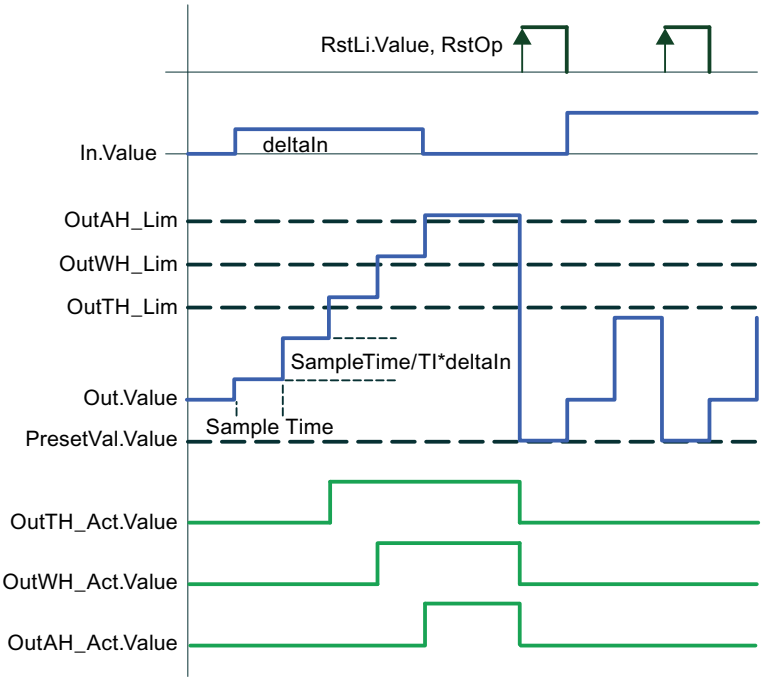
Using this parameterization, determine whether the block is to operate as an edge-triggered adder or summer, continuous adder or summer or as an integrator:

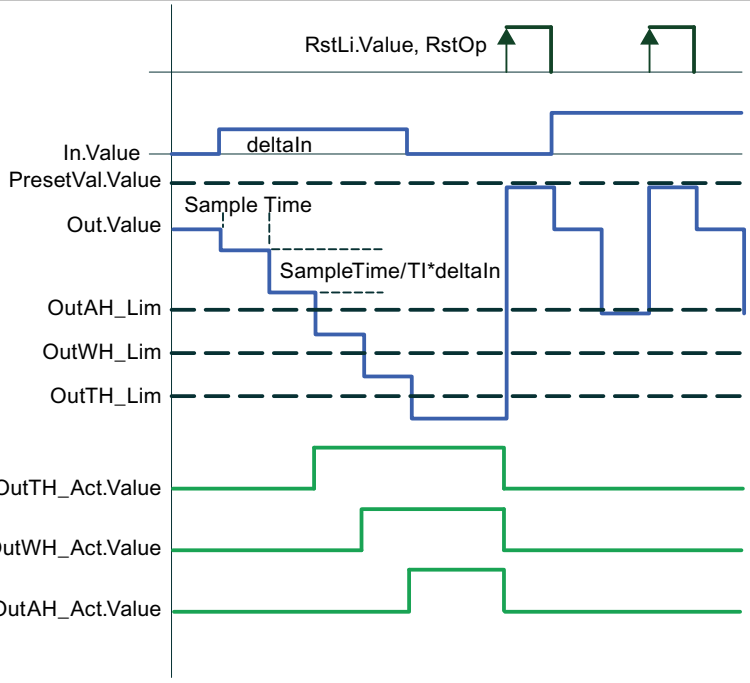
| Parameterization on the `Feature` Bit | How the block operates |
|---|---|
| `Feature.Bit6` = 0<br>`Feature.Bit7` = 0 | Block operates as an edge-triggered adder or summer.<br>The calculation at the output parameter `Out` is performed with a 0 - 1 edge at input parameter `P_In`.<br>The output parameter `Mode` shows whether the value is incremented or decremented.<br>The pulses counted at the input parameter `P_In` are output to the output parameter `CntOut`. |
| | Triggered sum up<br>$Out.Value = Out.Value_{(n-1)} + Incr.Value$<br> |
| | Triggered sum down<br>$Out.Value = Out.Value_{(n-1)} - Decr.Value$ |

| Parameterization on the `Feature` Bit | How the block operates |
|---|---|
| |  |

| Parameterization on the `Feature` Bit | How the block operates |
|---|---|
| `Feature.Bit6` = 0<br>`Feature.Bit7` = 1 | Block operates as a continuous adder or summer.<br>The calculation at the output parameter `Out` is performed continuously.<br>The output parameter `Mode` shows whether the value is incremented or decremented.<br>The result is output to the output parameter `Out`.<br>The output parameter `CntOut` always includes the value 0. |
| | Continuously sum up<br>$Out.Value = Out.Value_{(n-1)} + Incr.Value$<br> |
| | Continuously sum down<br>$Out.Value = Out.Value_{(n-1)} - Decr.Value$ |

| Parameterization on the `Feature` Bit | How the block operates |
|---|---|
| |  |

| Parameterization on the `Feature` Bit | How the block operates |
|---|---|
| `Feature.Bit6` = 1<br>`Feature.Bit7` = non-operational | Block operates as an continuous integrator.<br>The direction of integration depends on the operation in the faceplate or from the interconnection.<br>The input value `In` is integrated in accordance with the trapezoid rule dependent upon the counting direction according to the formula of the Integral block (Page 2253).<br>The output parameter `Mode` shows whether the value is integrated up or down.<br>The result is output to the output parameter `Out`.<br>The output parameter `CntOut` always includes the value 0. |
| | Continuously integrate up<br>`Mode` = 1: Out.Value = Out.Value n-1 + SampleTime/TI * (In.Value + In.Value n-1)/2<br>Set direction of integration:<br>• Operation in the faceplate using the "On" command (`UpOp` = 1)<br>Requirement: `LiOp.Value` = 0<br>• Interconnection `LiOp.Value` = 1 and `UpLi.Value` = 1<br> |
| | Continuously integrate down<br>`Mode` = 2: Out.Value = Out.Value n-1 - SampleTime/TI * (In.Value + In.Value n-1)/2<br>Set direction of integration:<br>• Operation in the faceplate using the "Off" command (`DnOp` = 1)<br>Requirement: `LiOp.Value` = 0<br>• Interconnection `LiOp.Value` = 1 and `DnLi.Value` = 1 |

| Parameterization on the `Feature` Bit | How the block operates |
|---|---|
| |  |



## Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

## Startup characteristics

Define the startup characteristics for this block via two `Feature` Bits:

Bit 0: Set startup characteristics (Page 148)

Bit 5: Use the last value following a complete download as the current value during startup of the block (Page 165)

The messages are suppressed after startup for the number of cycles specified in the `RunUpCyc` parameter.

## Time response

The block does not have any time response.

### Status word allocation for `Status1` parameter

You can find a description for each parameter in section TotalL I/Os (Page 2140).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | `SimOn` |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | `BypassAct.Value` |
| 6 | `OnAct.Value` |
| 7 - 8 | Not used |
| 9 | Display of `BypassAct.Value` in faceplate (display and operator controls) and block icon |
| 10 | `SimLiOp.Value` |
| 11 | `LiOp` |
| 12 - 13 | Not used |
| 14 | 1 = Invalid signal status |
| 15 | Triggered summing; `Feature.Bit6` = 0, `Feature.Bit7` = 0 |
| 16 | Continuous summing; `Feature.Bit6` = 0, `Feature.Bit7` = 1 |
| 17 | Integrate; `Feature.Bit6` = 1 |
| 18 - 31 | Not used |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | `MsgLock.Value` |
| 1 | `OutAH_Act.Value` |
| 2 | `OutWH_Act.Value` |
| 3 | `OutTH_Act.Value` |
| 4 | `OutTL_Act.Value` |
| 5 | `OutWL_Act.Value` |
| 6 | `OutAL_Act.Value` |
| 7 | `OutAH_En` OR `Feature.Bit23` |
| 8 | `OutWH_En` OR `Feature.Bit23` |
| 9 | `OutTH_En` OR `Feature.Bit23` |
| 10 | `OutTL_En` OR `Feature.Bit23` |
| 11 | `OutWL_En` OR `Feature.Bit23` |
| 12 | `OutAL_En` OR `Feature.Bit23` |
| 13 | `OutAH_MsgEn` |
| 14 | `OutWH_MsgEn` |
| 15 | `OutTH_MsgEn` |
| 16 | `OutTL_MsgEn` |
| 17 | `OutWL_MsgEn` |

| Status bit | Parameter |
|---|---|
| 18 | `OutAL_MsgEn` |
| 19 | Not used |
| 20 | `Mode` = 1, upward summing or integration |
| 21 | `Mode` = 0, summer/integrator off |
| 22 | `Mode` = 2, downward summing or integration |
| 23 - 28 | Not used |
| 29 | 1= Suppressed messages |
| 30 | Internal error |
| 31 | `MS_RelOp` |

### See also

TotalL operating modes (Page 2132)

TotalL functions (Page 2133)

TotalL error handling (Page 2138)

TotalL messaging (Page 2139)

TotalL block diagram (Page 2146)

## 11.3.2 TotalL operating modes

### TotalL operating modes

This block provides the following operating modes:

- On (Page 77)
- Out of service (Page 77)

### "On"

General information on the "On" mode is available in the section On (Page 77).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit4` =1 and `Feature.Bit4` =0 is as shown below:

| Output Parameters | **`Feature.Bit4` =0** | **`Feature.Bit4` =1** |
|---|---|---|
| Out.ST | 16#80 | 16#60 |
| OutAH_Act.ST | 16#XX | 16#60 |
| OutWH_Act.ST | 16#XX | 16#60 |

| Output Parameters | Feature.Bit4 =0 | Feature.Bit4 =1 |
|---|---|---|
| OutTH_Act.ST | 16#XX | 16#60 |
| OutTL_Act.ST | 16#XX | 16#60 |
| OutWL_Act.ST | 16#XX | 16#60 |
| OutAL_Act.ST | 16#XX | 16#60 |

**See also**

Description of TotalL (Page 2123)
TotalL functions (Page 2133)
TotalL error handling (Page 2138)
TotalL messaging (Page 2139)
TotalL I/Os (Page 2140)
TotalL block diagram (Page 2146)

## 11.3.3 TotalL functions

### Functions of TotalL

The functions for this block are listed below.

### Limit monitoring of the count value

This block provides the standard function Limit monitoring of the count value (Page 96).

### Group display SumMsgAct for limit monitoring, CSF and ExtMsgx

The block provides the standard function Group display for limit monitoring, CSF and ExtMsgx (Page 92).

### Suppressing messages using the MsgLock parameter

This block provides the standard function Suppressing messages using the MsgLock parameter (Page 231).

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

For `In` and `Out` different units of measure can be entered.

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

You can simulate the following values:

- Output value (`SimOut, SimOutLi`)

**Note**

**Note on the simulation of the output value `Out`**

For the presetting of the summing or integration value `Out` via the `RstLi` input, a 0 → 1 signal change at the input parameter `RstLi` is necessary during simulation.

### Bypass function

This block provides the standard function Bypassing signals (Page 115).

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is always formed from the following parameters:

- `P_In.ST`
- `Incr.ST`
- `Decr.ST`,
- `PresetVal.ST`
- `In.ST`
- `Out.ST`

The following signal status is formed by:

| Signal status | Used status |
|---|---|
| `Out.ST` | **Block operates as an edge-triggered adder or summer:**<br>Formed from the signal status `Incr.ST, Decr.ST, P_In.ST` and the last signal status `Out.St`.<br>**Block operates as continuous adder or summer:**<br>Formed from the signal status `Incr.ST, Decr.ST` and the last signal status `Out.St`.<br>**Block operates as continuous integrator:**<br>Formed from the signal status `In.ST` and the last signal status `Out.St`.<br>Resetting TotalL also resets the signal status: Out, ST := 16#80 |

With internal simulation, `ST_Worst` and the status of `Out` become `16#60`.

If there is a non-displayable floating-point number at one of the following parameters `In`, `PresetVal`, `TI` or `Out` when integrating `Feature.Bit6` = 1, the result is `ST_Worst` and the status of `Out` is set to `16#28`, provided the internally calculated status of `Out` does not result in `16#0` or the block is in internal simulation.

In addition, the status of `Out` is also written to the status of the output parameter `UpDnAct`.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` I/O in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 3 | Disable limit act output when limit value message is disabled (Page 209) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 4 | Changes signal status of outputs in OOS (Page 208) |
| 5 | Use the last value following a complete download as the current value during startup of the block (Page 165) |
| 6 | Block as summing unit or integrator (Page 154) |
| 7 | Summing characteristics continuous or triggered (Page 184) |
| 9 | Substitution value is active if the block is in bypass (Page 199) |
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 23 | Limits disabled in faceplate (Page 206) |
| 24 | Enabling local operator authorization (Page 169) |
| 25 | Suppression of all messages (Page 185) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 28 | Disabling operating points (Page 156) |
| 29 | Signaling limit violation (Page 181) |
| 30 | Set reset depending on the operating mode or the LiOp parameter (Page 174) |

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can stop the summing or integration |
| 5 | 1 = Operator can activate the upward summing or integration |
| 6 | 1 = Operator can activate the downward summing or integration |

| Bit | Function |
|---|---|
| 7 | 1 = Operator can preset the sum value (Out) |
| 8 - 9 | Not used |
| 10 | 1 =Operator can change the simulation value SimOut |
| 11 | 1 = Operator can activate the Simulation function |
| 12 | 1 = Operator can activate the Release for maintenance function |
| 13 | 1 = Operator can change the limit (OutAH_Lim) for high alarm |
| 14 | 1 = Operator can change the limit (OutWH_Lim) for high warning |
| 15 | 1 = Operator can change the limit (OutTH_Lim) for high tolerance |
| 16 | Not used |
| 17 | 1 = Operator can change the limit (OutAL_Lim) for low alarm |
| 18 | 1 = Operator can change the limit (OutWL_Lim) for low warning |
| 19 | 1 = Operator can change the limit (OutTL_Lim) for low tolerance |
| 20 | 1 = Operator can activate bypass functionality |
| 21 | 1 = Operator can deactivate bypass functionality |
| 22 | 1 =Operator can specify the default value (PresetVal.Value) |
| 23 | 1 = Operator can specify the integration time constant (TI) |
| 24 | 1 = Operator can specify the value for the increment (Incr.Value) |
| 25 | 1 = Operator can specify the value for the decrement (Decr.Value) |
| 26 | 1 = Operator can activate / deactivate messages via OutAH_MsgEn |
| 27 | 1 = Operator can activate / deactivate messages via OutAL_MsgEn |
| 28 | 1 = Operator can activate / deactivate messages via OutWH_MsgEn |
| 29 | 1 = Operator can activate / deactivate messages via OutTH_MsgEn |
| 30 | 1 = Operator can activate / deactivate messages via OutTL_MsgEn |
| 31 | 1 = Operator can activate / deactivate messages via OutWL_MsgEn |

The block has the following permissions for the OS1Perm parameter:

| Bit | Function |
|---|---|
| 0 - 15 | reserved |

**Note**

If you interconnect a parameter that is also listed in OS_Perm as a parameter, you have to reset the corresponding OS_Perm bit.

### Readback of the most recently calculated sum

As long as the output parameter Mode ≠ 0, the preceding sum value (OldOut) or the preceding count value (OldCntOut) is corrected to the current value:

- OldOut = Out.Value
- OldCntOut = CntOut.Value

After a warm restart, the sum value (Out) is automatically reset to the default value for the input parameter PresetVal if you have set the Feature Set startup characteristics (Page 148)Bit accordingly.

The count value CntOut in this case is always reset to 0.

The current sum and/or count value is set to the default predecessor values at startup via the Feature bit Use the last value following a complete download as the current value during startup of the block (Page 165).

- Old.Value = OldOut
- CntOut.Value = OldCntOut

### Setting the summing/integrating value to the default

PresetVal is used to specify a summing/integrating value that is used in a warm restart.

As long as RstLi or RstOp is set, the following applies:

Out.Value = PresetVal.Value.

When the input PresetVal is not interconnected (PresetVal.ST = FF), the default setting for the count value can also be made from the parameter view of the block.

RstLi or RstOp reset the CntOut output to 0 at the same time.

You can use the Feature.Bit30 "Set reset depending on operating mode or the LiOp parameter" to configure the setting of the count value to a default depending on the LiOp parameter:

Feature.Bit30 = 0: The reset does not depend on LiOp

Feature.Bit30 = 1: Setting to the default depends on LiOp

LiOp =0: Setting to the default PresetVal or resetting of CntOut can only take place via the faceplate or at the RstOp parameter for the faceplate.

LiOp =1: Setting to the default PresetVal or resetting of CntOut can only take place via the interconnectable RstLi input.

### Release for maintenance

This block provides the standard function Release for maintenance (Page 70).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

Description of TotalL (Page 2123)

TotalL operating modes (Page 2132)

TotalL error handling (Page 2138)

TotalL messaging (Page 2139)

TotalL I/Os (Page 2140)

TotalL block diagram (Page 2146)

## 11.3.4 TotalL error handling

### Error handling of TotalL

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 11 | The integration time constant `TI` is within the range:<br>`-SampleTime` / 2 < `TI` < S`ampleTime` / 2.<br>The integration is stopped. |
| 15 | `PresetVal.Value` > `OutOpScale.High`<br>`PresetVal.Value` < `OutOpScale.Low` |
| 30 | The value of `In.Value` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out.Value` output. |
| 31 | The value of `Out` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out` output. |
| 32 | The value of `PresetVal.Value` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out.Value` output. |
| 33 | The value of `TI` can no longer be displayed in the REAL number field. The integration is stopped. |
| 34 | The value `Incr.Value` or `Decr.Value` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out.Value` output. |
| 51 | Invalid signal at `LiOp` = 1:<br>`OffLi` = 1 and `UpLi` = 1 and/or `DnLi` = 1<br>`OffLi` = 0 and `UpLi` = 1 and `DnLi` = 1 |

### See also

Description of TotalL (Page 2123)

TotalL operating modes (Page 2132)

TotalL functions (Page 2133)

TotalL messaging (Page 2139)

TotalL block diagram (Page 2146)

TotalL I/Os (Page 2140)

## 11.3.5 TotalL messaging

### Messaging

The following messages can be generated for this block:

- Functions for displaying measured limits

Messages generated as a reaction to limit violations, can be suppressed by the settings `MsgEn` and `MsgLock`.

### Associated values for message instance `MsgEvId`

| Associated value | Block parameters |
|---|---|
| 1 | `BatchName` |
| 2 | `StepNo` |
| 3 | `BatchID` |
| 4 | `ExtVa104` |
| 5 | `ExtVa105` |

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId` | SIG 1 | Alarm - high | $$BlockComment$$<br>High alarm limit violated |
| | SIG 2 | Warning - high | $$BlockComment$$<br>High warning limit violated |
| | SIG 3 | Tolerance - high | $$BlockComment$$<br>High tolerance limit violated |
| | SIG 4 | Tolerance - low | $$BlockComment$$<br>Low tolerance limit violated |
| | SIG 5 | Warning - low | $$BlockComment$$<br>Low warning limit violated |
| | SIG 6 | Alarm - low | $$BlockComment$$<br>Low alarm limit violated |
| | SIG 7 | AS process control message - fault | $$BlockComment$$<br>External message 1 |
| | SIG 8 | AS process control message - fault | $$BlockComment$$<br>External message 2 |

Explanation:

$$BlockComment$$: Content of the instance-specific comment

### See also

Description of TotalL (Page 2123)
TotalL operating modes (Page 2132)
TotalL functions (Page 2133)
TotalL error handling (Page 2138)
TotalL I/Os (Page 2140)
TotalL block diagram (Page 2146)

## 11.3.6 TotalL I/Os

TotalL I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| BypLiOp | 1 = Bypass commands via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypLock | 1 = Bypass activation or deactivation is locked for operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypOut | Substitution value if block is in bypass | REAL | 0.0 |
| BypOutLi | 1 = Select bypass Out (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| BypOutOp | 1 = Select bypass Out (via operator) | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Decr | Decrement for downward summing<br>Decr.ST = FF via operator<br>Decr.ST <> FF via interconnection | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| DnLi* | 1 = Down counter, via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DnOp* | 1 = Down counter, via operator | BOOL | 0 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| ExtMsg1 | Binary input for freely selectable message 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtMsg2 | Binary input for freely selectable message 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtVa104 | Associated value 4 for messages (MsgEvId) | ANY | |
| ExtVa105 | Associated value 5 for messages (MsgEvId) | ANY | |
| Feature | I/O for additional functions (Page 2133) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| In | Input for integral value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Incr | Increment for upward summing<br>Incr.ST = FF via operator<br>Incr.ST <> FF via interconnection | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| InUnit | Unit of measure for the input parameter In | INT | 1351 |
| LiOp | 1 = Interconnection<br>0 = Operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MsgEvId | Message number (assigned automatically) | DWORD | 16#00000000 |
| MsgLock | 1 = Suppress process messages.<br>See also the section Suppressing messages using the MsgLock parameter (Page 231) for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_RelOp* | Operator can switch release for maintenance | BOOL | 0 |
| Occupied | 1 = In use by a batch | BOOL | 0 |
| OffLi* | 1 = Counter disabled via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OffOp* | 1 = Counter disabled via operator | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OldCntOut* | Previous count value | DINT | 0 |
| OldOut* | Previous output value | REAL | 0.0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operating permissions (Page 2133) | STRUCT<br>• Bit 0: BOOL<br>• Bit 10: BOOL<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| OS1Perm | I/O for operator permissions (Page 2133) | STRUCT<br>• Bit 0: BOOL<br>• Bit 15: BOOL | -<br>• 1<br>• 1 |
| OutAH_En | 1 = Enable alarm (high) for count | BOOL | 1 |
| OutAH_Lim | Limit for count alarm (high) | REAL | 95.0 |
| OutAH_MsgEn | 1 = Enable message for count alarm (high) | BOOL | 1 |
| OutAL_En | 1 = Enable alarm (low) for count | BOOL | 1 |
| OutAL_Lim | Limit for count alarm (low) | REAL | 0.0 |
| OutAL_MsgEn | 1 = Enable message for count alarm (low) | BOOL | 1 |
| OutOpScale | Limit for scale in Out bar graph of faceplate | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| OutTH_En | 1 = Enable tolerance message (high) | BOOL | 0 |
| OutTH_Lim | Count tolerance limit message (high) | REAL | 85.0 |
| OutTH_MsgEn | 1 = Enable message for count tolerance message (high) | BOOL | 1 |
| OutTL_En | 1 = Enable count tolerance message (low) | BOOL | 0 |
| OutTL_Lim | Count tolerance message limit (low) | REAL | 0.0 |
| OutTL_MsgEn | 1 = Enable message for count tolerance message (low) | BOOL | 1 |
| OutUnit | Unit of measure for count Out | INT | 1038 |
| OutWH_En | 1 = Enable count warning (high) | BOOL | 1 |
| OutWH_Lim | Count warning limit (high) | REAL | 90.0 |
| OutWH_MsgEn | 1 = Enable message for count warning (high) | BOOL | 1 |
| OutWL_En | 1 = Enable count warning (low) | BOOL | 1 |
| OutWL_Lim | Count warning limit (low) | REAL | 0.0 |
| OutWL_MsgEn | 1 = Enable message for count warning (low) | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| P_In | Input for sum value (pulse input) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PresetVal* | Default setting for the count | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| RstBypLi | 1 = Reset bypass Out (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstBypOp | 1 = Reset bypass Out (via operator) | BOOL | 0 |
| RstLi | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstOp* | 1 = Reset via operator | BOOL | 0 |
| RunUpCyc | Messages can be suppressed for a short time after startup. You can set the number of cycles using the input parameter RunUpCyc | INT | 3 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | |
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | |
| SimLiOp | Activation/deactivation of the simulation by:<br>0 = Operator<br>1 = Interconnection or SFC | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOnLi | 1 = Simulation via interconnection or SFC (controlled by SimLiOp = 1) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn* | 1 = Simulation on | BOOL | 0 |
| SimOut* | Output value used for SimOn = 1 | REAL | 0.0 |
| SimOutLi | Output value that is used for SimOnLi.Value = 1 (SimLiOp.Value = 1) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| TI* | Intagral component time | REAL | 1.0 |
| UpLi* | 1 = Forward counter, via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| UpOp* | 1 = Forward counter, via operator | BOOL | 0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| BypassAct | 1 = Bypass is activated in this block | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CntOut | Number of counted pulses | DINT | 0 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see TotalL error handling (Page 2138) | INT | -1 |
| Mode | 0 = Totalizer off<br>1 = Totalizer summed/integrated upward<br>2 = Totalizer summed/integrated downward | INT | 0 |
| MsgAckn | Message acknowledgment status (output ACK_STATE of ALARM_8P) | WORD | 16#0000 |
| MsgErr | 1 = Message error (output ERROR of ALARM_8P) | BOOL | 0 |
| MsgStat | Message status (output STATUS of ALARM_8P) | WORD | 16#0000 |
| MS_Release | Release for maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| O_MS_Ext | Reserved | DWORD | 0 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feed-forwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| OS1PermOut | Display of OS1_Perm | WORD | 16#FFFF |
| OS1PermLog | Display of OS1_Perm with settings changed by the block algorithm | WORD | 16#FFFF |
| Out | Count | STRUCT<br>• Value: REAL<br>• ST: BYTE<br>• Bit 1: BOOL of internal fill byte (Bypass information bit) | -<br>• 0.0<br>• 16#80<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OutAH_Act | 1 = Count alarm (high) enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| OutAL_Act | 1 = Count alarm (low) enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| OutOpScaleOut | OutOpScale as output | • High: REAL<br>• Low: REAL | • 100.0<br>• 0.0 |
| OutTH_Act | 1 = Count tolerance message (high) enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| OutTL_Act | 1 = Count tolerance message (low) enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| OutUnitOut | OutUnit as output | INT | 0 |
| OutWH_Act | 1 = Count warning (high) enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| OutWL_Act | 1 = Count warning (low) enabled.<br>You can change the reaction for this parameter with Feature.Bit28 (Disabling operating points (Page 156)) and with Feature.Bit29 (Signaling limit violation (Page 181)). | STRUCT<br>• Value: BOOL<br>• ST: BYTE<br>• Bit 1: BOOL (Bypass information bit) | -<br>• 0<br>• 16#80<br>• 0 |
| SimAct | 1 = Simulation Active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 2123) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 2123) | DWORD | 16#00000000 |
| UpDnAct | 1 = Block counts/integrates<br>0 = Block has no activity | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

**See also**

TotalL operating modes (Page 2132)
TotalL messaging (Page 2139)
TotalL block diagram (Page 2146)

## 11.3.7 TotalL block diagram

### TotalL block diagram

A block diagram is not provided for this block.

**See also**

Description of TotalL (Page 2123)
TotalL operating modes (Page 2132)
TotalL functions (Page 2133)
TotalL error handling (Page 2138)
TotalL I/Os (Page 2140)
TotalL messaging (Page 2139)

## 11.3.8 Operator control and monitoring

### 11.3.8.1 TotalL views

### Views of the TotalL block

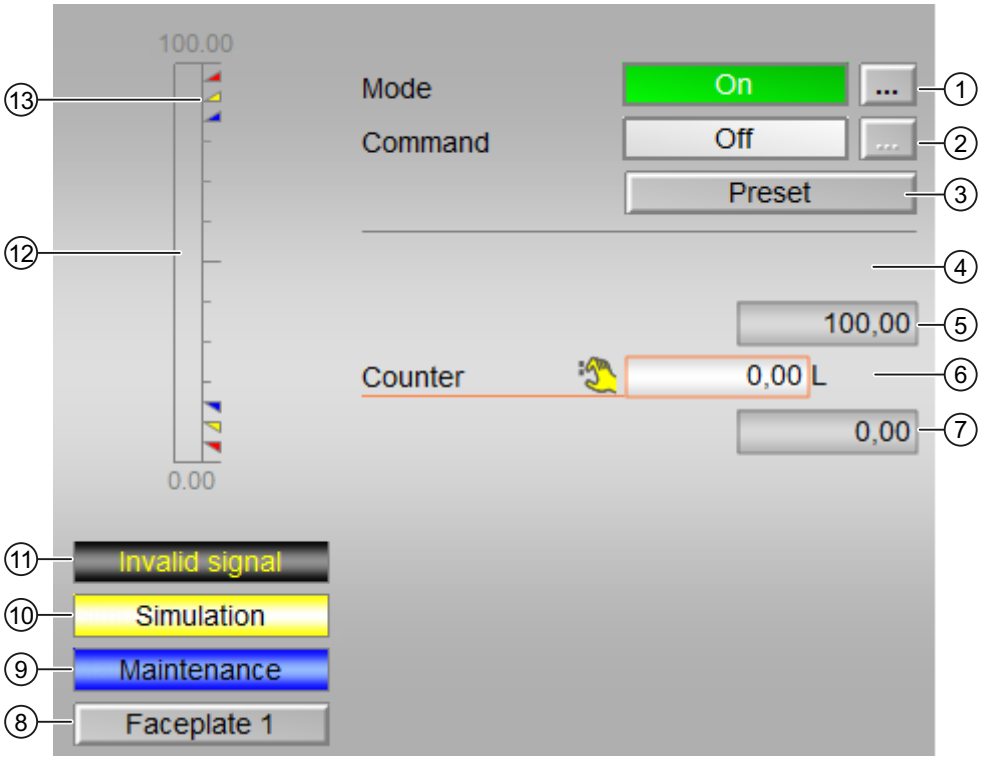
The block TotalL provides the following views:

- TotalL standard view (Page 2147)
- Alarm view (Page 336)
- TotalL limit view (Page 2150)
- Trend view (Page 341)
- TotalL parameter view (Page 2151)
- TotalL preview (Page 2153)
- Memo view (Page 338)
- TotalL block icon (Page 2154)
- Batch view (Page 336)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 11.3.8.2 TotalL standard view

#### TotalL standard view




#### (1) Displaying and switching the operating mode

Operation of inputs `OosOp` and `OnOp`.

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Bar graph (Page 376): `Status1.Bit3` (`OosAct.Value`)
- Out of service (Page 77): `Status1.Bit6` (`OnAct.Value`)

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the operating mode.

#### (2) Enabling and disabling the counter

Operation of the inputs `UpOp`, `OffOp` and `DnOp`.

This area shows you the default operating state for the counter. The following states can be shown and executed here:

- "On ⇑": `Status2.Bit20` (`Mode` = 1)
- "On ⇓": `Status2.Bit22` (`Mode` = 2)
- "Off": `Status2.Bit21` (`Mode` = 0)

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

### (3) Default setting

Operation of the input `RstOp`.

This button activates the preset value.

### (4) Display of the count value

The current count values are displayed here:

**Display of input In**

Depending on `Feature.Bit6` and `Feature.Bit7`, either input `In` or an empty frame is displayed:

- "Input": `Feature.Bit6` = 1, otherwise no display

The display is not operational. Format of `In` like display `Out`.

### (5) High scale range for the count value

This value provides information on the display range for the bar graph (above) of the count value. The scale range is defined in the engineering system.

### (6) Display of the count value

The current count values are displayed here:

**Display Out**

Depending on `Feature.Bit6`, the text for the display is switched over:

- "Counter": `Feature.Bit6` = 0
- "Integrator": `Feature.Bit6` = 1

The "Counter" display is not operational, not even in simulation. The format of `In` is like the `Out` display.

The "Integrator" display is only operational in simulation (operation input `SimOut`). The format corresponds to the block icon of `AnalogValueFormat1`.

### (7) Low scale range for the count value

This value provides information on the display range for the bar graph (below) of the count value. The scale range is defined in the engineering system.

### (8) Navigation button for switching to the standard view of any faceplate

Use this navigation button to open the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234).

### (9) Display area for block states

This area provides additional information on the operating state of the block:

- "Maintenance"

You can find more information about this in the section Release for maintenance (Page 70).

### (10) Display area for block states

This area provides additional information on the operating state of the block:

- "Simulation"

You can find additional information on this in the section Simulating signals (Page 64).

### (11) Display area for block states

This area provides additional information on the operating state of the block:

- "Internal error"
- "Invalid Signal"

You can find additional information on this in the section Error handling (Page 128).

### (12) Graphic display of the current count value

This area shows you the current count in form of a bar graph. You can also see the high and low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES). For more information refer Bar graph (Page 376) section.

### (13) Limit display

These colored triangles indicate the configured limits in the respective bar graph:

- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

In counting mode "count up", only the colored triangles of the upper limits are visible; and in counting mode "count down", only the colored triangles of the lower limits are visible.

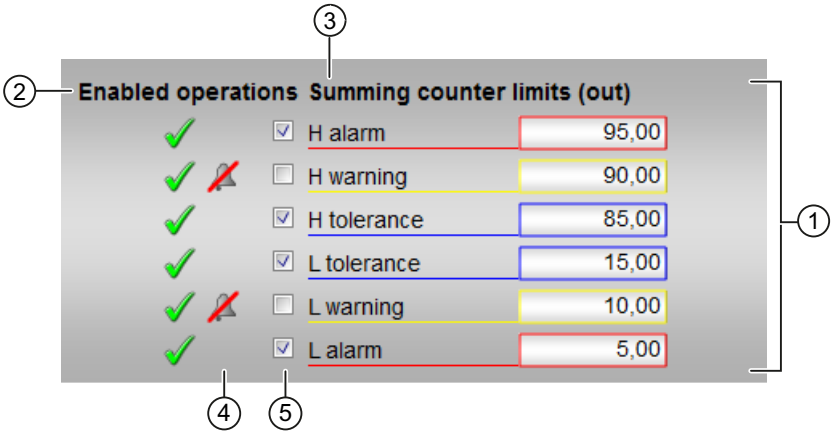
For more information, refer: Bar graph (Page 376)

### See also

On (Page 77)

## 11.3.8.3 TotalL limit view

### Limit view of TotalL




### (1) Limits for the counter

In this area, you can enter the limits for the counter. Refer to the Changing values (Page 291) section for more on this.

You can change the following limits:

- "H alarm": Alarm high
- "H warning": Warning high
- "H tolerance": Tolerance high
- "L tolerance": Tolerance low
- "L warning": Warning low
- "L alarm": Alarm low

Format and unit is like the `Out` display in the standard view.

### (2) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

### (3) Dynamic text

Depending on `Feature.Bit6`, the text is switched:

- "Integrator limits (out)": Display for `Feature.Bit6` = 1
- "Summer limits (out)": Display for `Feature.Bit6` = 0

### (4) "Message suppression"

Message suppression indicates whether or not the suppression of the associated message in the AS block is activated with the `xx_MsgEn` parameter. The output of messages is not suppressed when the block is installed (all `xx_MsgEn` parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

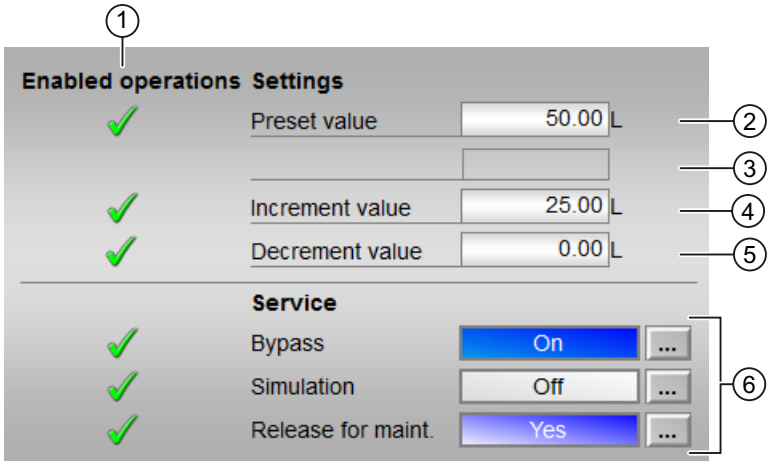
### (5) Suppress messages

You can enable / disable messages by setting the check mark.

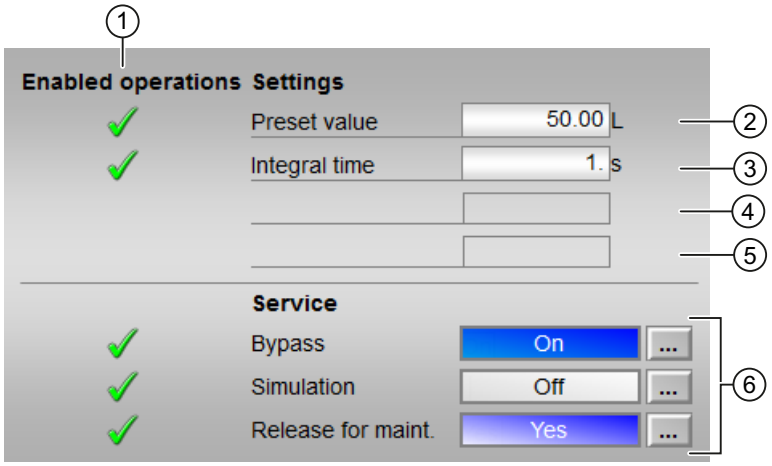
## 11.3.8.4 TotalL parameter view

### Parameter view of TotalL

With `Feature.Bit6` = 0:




With `Feature.Bit6` = 1:




### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`).

### (2) Default value

In this section you can enter the value from which the counting should begin. Additional information is available in the section Changing values (Page 291).

### (3) Integral time

The currently integral time in seconds is displayed in this area.

- "Integral time" Display for `Feature.Bit6` = 1, otherwise empty.

### (4) Increment value

The increment value is displayed in this area.

- "Increment value": Display for `Feature.Bit6` = 0, otherwise empty.

### (5) Decrement value

The decrement value is displayed in this area.

- "Decrement value": Display for `Feature.Bit6` = 0, otherwise empty.

### (6) Service

You can select the following function in this area:

- "Bypass"
- "Simulation"
- "Release for maintenance"

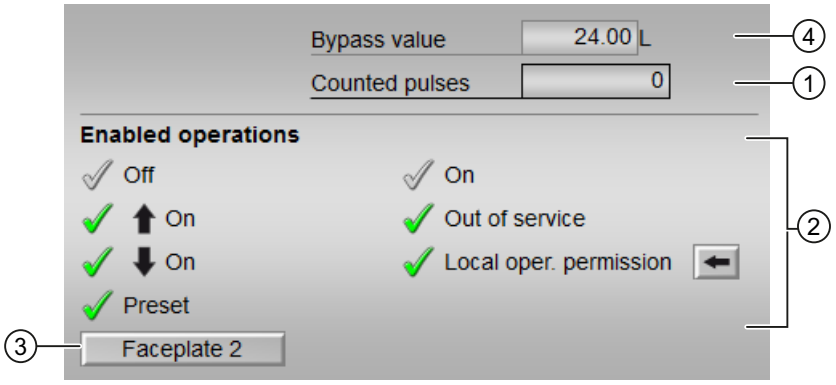
Refer to the Switching operating states and operating modes (Page 289) section for more on this.

You can find information on this area in the section:

- Bypassing signals (Page 115)
- Simulating signals (Page 64)
- Release for maintenance (Page 70)

## 11.3.8.5 TotalL preview

### Preview of TotalL




### (1) Counted pulses

The number of pulses already counted is displayed in this area.

- "Counted pulses": `Feature.Bit6` = 0 and `Feature.Bit7` = 0, otherwise empty

### (2) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "Off": You can disable the counter.
- "On ⇑": You can operate the incremental counter.
- "On ⇓": You can operate the decremental counter.
- "Default": You can change the default setting.
- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. Additional information is available in the section Operator control permissions (Page 285).

### (3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

Additional information is available in the section Opening additional faceplates (Page 234).

### (4) Bypass value

This area displays the bypass value (`BypOut`).

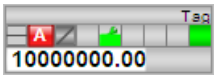
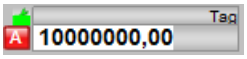
## 11.3.8.6 TotalL block icon

### Block icons for TotalL

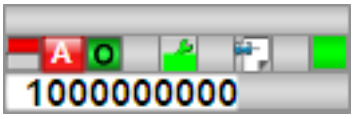
A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits
- Operating modes
- Signal status, release for maintenance
- Memo display
- Display counter running

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | Block icon in the full display |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269).

# 11.4 CntOhSc - Runtime determination and counters with counting direction "up"

## 11.4.1 Description of CntOhSc

### Object name (type + number) and family

Type + number: FB 1803

Family: Count

### Area of application of CntOhSc

The block is used for the following applications:

- Incrementing of operating hours
- Incrementing a defined input value

### How it works

The block determines the time in which a unit has been in operation and it counts a defined input value. The block can only count forwards.

1. Off (`OffOp = 1`)
   The block is disabled (output parameter `CountMode = 0`). No counting takes place.
2. Increment (`UpOp = 1`)
   The operating time of the connected unit is incremented (output parameter `CountMode = 1`).

The operating time is shown in days, hours, minutes and seconds. The maximum operating time is 32767 days and 23 hours.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is first installed automatically in the startup OB (OB 100).

### Startup characteristics

Define the startup characteristics for this block via two `Feature` bits:

Bit 0: Set startup characteristics (Page 148)

Bit 5: Use the last value following a complete download as the current value during startup of the block (Page 165)

When you set `Feature.Bit5` to 1, then:

- `Days:= OldDays`
- `Hours:= OldHours`

- `Minutes:= OldMinutes`
- `Seconds:= OldSeconds`

### Status word allocation for `Status1` parameter

Refer to the following section for a description of the individual parameters: CntOhSc I/Os (Page 2161)

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | Not used |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | Not used |
| 6 | `OnAct.Value` |
| 7 - 10 | Not used |
| 11 | `LiOp` |
| 12 - 13 | Not used |
| 14 | 1 = Invalid signal status |
| 15 | Not used |
| 16 | Internal error |
| 17 - 31 | Not used |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | Not used |
| 1 | `HrsHiL1Act.Value` |
| 2 | `HrsHiL2Act.Value` |
| 3 | `CntHiL1Act.Value` |
| 4 | `CntHiL2Act.Value` |
| 5 - 6 | Not used |
| 7 | `HrsHiL1En` OR `Feature.Bit23` |
| 8 | `HrsHiL2En` OR `Feature.Bit23` |
| 9 | `CntHiL1En` OR `Feature.Bit23` |
| 10 | `CntHiL2En` OR `Feature.Bit23` |
| 11 - 19 | Not used |
| 20 | Count up |
| 21 | Counter off |
| 22 - 31 | Not used |

## 11.4.2 CntOhSc operating modes

### Operating modes of CntOhSc

The block can be operated using the following modes:

- On (Page 77)
- Out of service (Page 77)

### "On"

General information on the "On" mode is available in the section On (Page 77).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit4` =1 and `Feature.Bit4` =0 is as shown below:

| Output Parameters | **`Feature.Bit4` =0** | **`Feature.Bit4` =1** |
|---|---|---|
| DeviceOn.ST | 16#XX | 16#60 |
| HrsHiL1Act.ST | 16#XX | 16#60 |
| HrsHiL2Act.ST | 16#XX | 16#60 |
| CntHiL1Act.ST | 16#XX | 16#60 |
| CntHiL2Act.ST | 16#XX | 16#60 |

## 11.4.3 CntOhSc functions

### Functions of CntOhSc

The block provides the following functions:

### Reset counter to zero

The output parameters of the operating time `Days`, `Hours`, `Minutes`, `Seconds`, `TimeHours` and `TotalTime` are reset via the interconnect `ResetOh` parameter. The reset is made with a 0 - 1 edge.q

The counted value at the `Cnt` output parameter is reset with the interconnectable `ResetCnt` parameter. The reset is made with a 0 - 1 edge.

The count value and the output parameters of the operating time can also be reset to zero together in the standard view at the `ResetOp` parameter or the interconnectable `ResetLi` parameter via the faceplate.

You can use `Feature.Bit30` "Set reset depending on operating mode or the `LiOp` parameter" to configure a reset depending on the `LiOp` parameter:

`Feature.Bit30` = 0: The reset does not depend on `LiOp`

`Feature.Bit30` = 1: The reset depends on `LiOp`

`LiOp =0`: Separate reset via inputs `ResetOh` and `ResetCnt` is not possible. Simultaneous reset can only be made via the faceplate or at the `ResetOp` parameter.

`LiOp =1`: Separate reset is only possible via interconnectable inputs `ResetOh` and `ResetCnt`. Simultaneous reset can only be made via the interconnectable input `ResetLi`.

### Configurable reactions using the Feature parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 4 | Changes signal status of outputs in OOS (Page 208) |
| 5 | Use the last value following a complete download as the current value during startup of the block (Page 165) |
| 23 | Limits disabled in faceplate (Page 206) |
| 24 | Enabling local operator authorization (Page 169) |
| 26 | Reaction of the switching points in the "Out of service" operating mode (Page 188) |
| 29 | Signaling limit violation (Page 181) |
| 30 | Set reset depending on the operating mode or the LiOp parameter (Page 174) |

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can stop counting |
| 5 | 1 = Operator can switch to "Count up" mode |
| 6 | 1 = Operator can reset the count |
| 7 - 12 | Not used |
| 13 | 1 = Operator can change the operating hours high limit 1 (`HrsHi1Lim`) |
| 14 | 1 = Operator can change the operating hours high limit 2 (`HrsHi2Lim`) |
| 15 | 1 = Operator can change the counter high limit 1 (`CntHi1Lim`) |

| Bit | Function |
|---|---|
| 16 | 1 = Operator can change the counter high limit 2 (`CntHi2Lim`) |
| 17 - 31 | Not used |

### Providing limit at the output

For further connections to other blocks the following input parameters will also be displayed on the output:

`CntHi1LimOut` := `CntHi1Lim`;

`CntHi2LimOut` := `CntHi2Lim`;

`HrsHi1LimOut` := `HrsHi1Lim`

`HrsHi2LimOut` := `HrsHi2Lim`;

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `InOh.ST`
- `InCnt.ST`
- `TmHoursST`
- `CntST`

The following signal status is formed by:

| Signal status | Used status |
|---|---|
| `TmHoursST` | Increment: Formed from the worst signal status of `InOh.ST` and the last signal status `TmHoursST`. This means that the worst signal status of `InOh.ST` is stored to `TmHoursST` during increment.<br>Counter off: Signal status `TmHoursST` is frozen.<br>Resetting `CntOhSc` or `ResetOh` also resets the signal status: `TmHoursST` := 16#80 |
| `CntST` | Increment: Formed from the worst signal status of `InCnt.ST` and the last signal status `CntST`. This means that the worst signal status of `InCnt.ST` is stored to `CntST` during increment.<br>Counter off: Signal status `CntST` is frozen.<br>Resetting `CntOhSc` or `ResetCnt` also resets the signal status: `CntST` := 16#80 |

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

## 11.4.4 CntOhSc error handling

### Error handling of CntOhSc

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; block is not processed |
| 0 | No active fault |
| 51 | Invalid signal at `LiOp = 1`:<br>• `OffLi = 1` and `UpLi = 1` |

## 11.4.5 CntOhSc messaging

### Messaging

This block does not offer messaging.

## 11.4.6 CntOhSc I/Os

I/Os of CntOhSc

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `BatchEn` | 1 = Enable allocation | BOOL | 0 |
| `BatchID` | Batch ID | DWORD | 16#00000000 |
| `BatchName` | Batch name | S7-String | |

| Parameter | Description | Type | Default |
|---|---|---|---|
| BlkTypeInfo* | Block Type Version and FB Number of block-type<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| CntHiL1En | 1 = Counter high limit 1 is enabled | BOOL | 0 |
| CntHiL2En | 1 = Counter high limit 2 is enabled | BOOL | 0 |
| CntHi1Lim | Counter high limit 1 | DINT | 95 |
| CntHi2Lim | Counter high limit 2 | DINT | 90 |
| CntOpHiScale | High limit for scale in count bar graph of face-plate | DINT | 100 |
| CntOpLoScale | Low limit for scale in count bar graph of face-plate | DINT | 0 |
| CntUnit | Unit of measure for count Cnt | INT | 0 |
| Feature | I/O for additional functions (Page 2158) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| HrsHiL1En | 1 = Operating hours high limit 1 is enabled | BOOL | 0 |
| HrsHiL2En | 1 = Operating hours high limit 2 is enabled | BOOL | 0 |
| HrsHi1Lim | Operating hours high limit 1 | DINT | 2280 |
| HrsHi2Lim | Operating hours high limit 2 | DINT | 2160 |
| HrsOpHiScale | Operating hours high limit of bar display for OS | DINT | 2400 |
| HrsOpLoScale | Operating hours low limit of bar display for OS | DINT | 0 |
| InOh | Digital input value for operating hours counter | REAL | 0.0 |
| InCnt | Digital input value for up counter | REAL | 0.0 |
| LiOp | 1= Interconnection<br>0 = Operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OffLi | Counter disabled via interconnection:<br>1 = Off | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OffOp* | Counter disabled via operator:<br>1 = Off | BOOL | 1 |
| OldCnt* | Previous count value | DINT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OldDays | Previous day value | INT | 0 |
| OldHours | Previous hour value | INT | 0 |
| OldMinutes | Previous minute value | INT | 0 |
| OldSeconds* | Previous second value | INT | 0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via operator | BOOL | 0 |
| OpSt_In | Input parameter for local operator authorization, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for CntOhSc functions (Page 2158) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| ResetCnt | 1 = Input valueCnt reset | BOOL | 0 |
| ResetLi | 1 = Counter reset via interconnection | BOOL | 0 |
| ResetOh | 1 = Operating hours reset | BOOL | 0 |
| ResetOp* | 1 = Counter reset by operator | BOOL | 0 |
| SampleTime | Sampling time [s] | REAL | 0.1 |
| SelFp1 | Open faceplate 1 | ANY | |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UpLi | 1 = Increment (via interconnection) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| UpOp* | 1 = Increment (via faceplate) | BOOL | 0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| CountMode | Count mode:<br>0 = Off<br>1 = Increment | INT | 0 |
| Cnt | Count | DINT | 0 |
| CntHiL1Act | 1 = Counter high limit 1 is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| CntHiL2Act | 1 = Counter high limit 2 is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CntHi1LimOut | CntHi1LimOut as output | DINT | 0 |
| CntHi2LimOut | CntHi2LimOut as output | DINT | 0 |
| CntST | Signal status for Cnt | BYTE | 16#80 |
| Days | Operating stage | INT | 0 |
| DeviceOn | 1 = Unit on | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see CntOhSc error handling (Page 2161) | INT | -1 |
| Hours | Operating hours | INT | 0 |
| HrsHiL1Act | 1 = Operating hours high limit 1 is active | BOOL | 0 |
| HrsHiL2Act | 1 = Operating hours high limit 2 is active | BOOL | 0 |
| HrsHi1LimOut | HrsHi1Lim as output | DINT | 0 |
| HrsHi2LimOut | HrsHi2Lim as output | DINT | 0 |
| Minutes | Operating minutes | INT | 0 |
| O_MS_Ext | Reserved | DWORD | 16#00000000 |
| OnAct | Block running | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | Block is "out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Operator permission: Output for OS | DWORD | 16#FFFFFFFF |
| OS_PermOut | Operator permission: Output for OS | DWORD | 16#FFFFFFFF |
| Seconds | Operating time [in sec] | INT | 0 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word 1 (Page 457) | DWORD | 16#00000000 |
| Status2 | Status word 2 (Page 457) | DWORD | 16#00000000 |
| TimeHours | Service life [h] | DINT | 0 |
| TmHoursST | Signal status for TimeHours | BYTE | 16#80 |
| TotalTime | Total operating time [in sec] | DWORD | 16#00000000 |

## 11.4.7 CntOhSc block diagram

### Block diagram of CntOhSc

A block diagram is not provided for this block.

## 11.4.8 Operator control and monitoring

### 11.4.8.1 CntOhSc views

### Views of the CntOhSc block

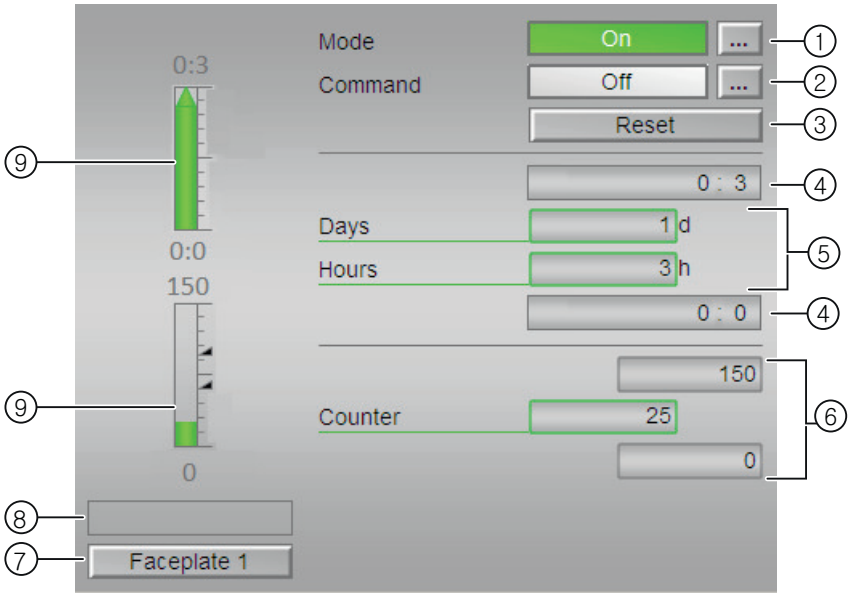
The CntOhSc block provides the following views:

- CntOhSc standard view (Page 2166)
- CntOhSc limit view (Page 2168)
- Trend view (Page 341)
- CntOhSc preview (Page 2169)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for CntOhSc (Page 2170)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 11.4.8.2 CntOhSc standard view

#### Standard view of CntOhSc




#### (1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 77)
- Out of service (Page 77)

Refer to section Switching operating states and operating modes (Page 289) for information on switching the operating mode.

#### (2) Enable and disable the counter

This area shows you the default operating state for the counter. The following states can be shown and executed here:

- "On ⇑"
- "Off"

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

#### (3) Reset

Operation of the input `RstOp`.

This button activates the preset value.

### (4) High and low scale range for the count value

These values provide information on the display range for the bar graph of the count `HrsOpHiScale` or `HrsOpLoScale`. The scale range is defined in the engineering system.

### (5) Display for counts

The following counts are shown here:

- "Days" with signal status `TmHoursST`
- "Hours" with signal status `TmHoursST`

### (6) Counter

The following values are shown here:

- Current count value with the signal status `CntST`
- High and low scale range for the count

These values provide information on the display range for the bar graph of the count `CntOpHiScale` or `CntOpLoScale`. The scale range is defined in the engineering system.

### (7) Navigation button for switching to the standard view of any faceplate

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234).

### (8) Display area for states of the block

This area provides additional information on the operating state of the block:

- "Internal error"
- "Invalid Signal"

You can find additional information on this in the section Error handling (Page 128).

### (9) Graphic display for the current count

These areas show the `Cnt` and `TimeHours` counts in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

For more information, refer: Bar graph (Page 376)

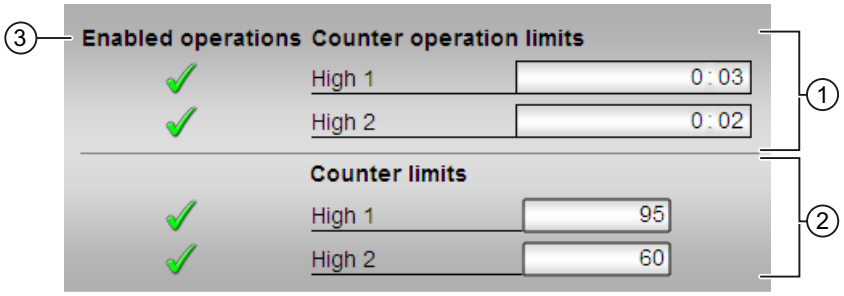
### Limits

These colored triangles indicate the configured limits in the respective bar graph:

- Red: High limit 1
- Yellow: High limit 2

## 11.4.8.3 CntOhSc limit view

### Limit view of CntOhSc




### (1) Operating hours limits

In this area, you can enter the limits for the operating hours. You can find additional information on this in the section Changing values (Page 291) .

You can change the following limits:

- "High 1": Operating hours high limit 1
- "High 2": Operating hours high limit 2

### (2) Counter limits

In this area, you can enter the limits for the counter. You can find additional information on this in the section Changing values (Page 291) .

You can change the following limits:

- "High 1": Counter high limit 1
- "High 2": Counter high limit 2

### (3) Enabled operation

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

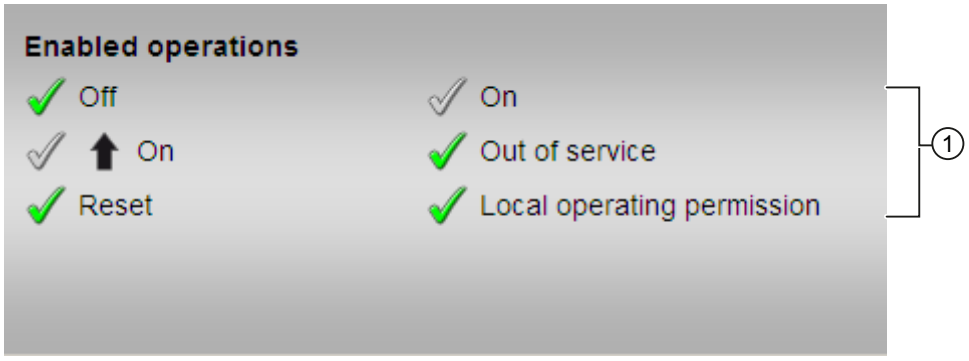
- **Green check mark:** The OS operator can control this parameter.
- **Gray check mark:** The OS operator cannot control this parameter at this time due to the process.
- **Red cross:** The OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm`).

### (4) "Message suppression"

Message suppression indicates whether or not the suppression of the associated message in the AS block is activated with the `xx_MsgEn` parameter. The output of messages is not suppressed when the block is installed (all `xx_MsgEn` parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

## 11.4.8.4 CntOhSc preview

### Preview of CntOhSc




### (1) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark:** the OS operator can control this parameter
- **Gray check mark:** the OS operator cannot control this parameter at this time due to the process
- **Red cross:** the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm`)

The following enabled operations are shown here:

- "Off": You can disable the counter.
- "On ⇑": You can operate the counter.
- "Reset": You can reset the counter after interlocks or errors.
- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. Additional information is available in the section Operator control permissions (Page 285).

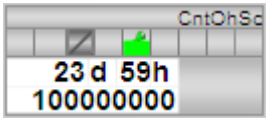
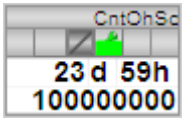
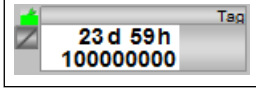
### 11.4.8.5 Block icon for CntOhSc

#### Block icons for CntOhSc

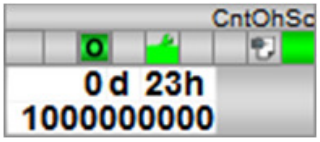
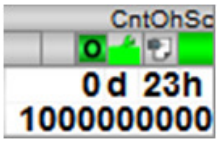
A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of the limits
- Operating modes
- Signal status
- Memo display
- Display counter running

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in the CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | Block icon in the full display |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in the CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | - | Block icon in "Out of service" mode (example with type block icon type 1) |

Additional information on the block icon and the control options in the block icon is available in the following chapters:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269).

# Timers 12

## 12.1 TimerP - Time delays signal forwarding / pulse generator

### 12.1.1 Description of TimerP

#### Object name (type+number) and family

Type + number: FB 1810

Family: TIME

#### Area of application for TimerP

The block is used for the following fields of applications:

- Pulse generator
- Extended pulse
- ON delay
- ON delay with memory
- OFF delay

#### How it works

The TimerP block is used for delayed forwarding of start or stop actions through the `Out` output parameter.

New values at the `Ti` I/O are only applied when a change is made to the `In` I/O.

The inverted signal for `Out` is also provided at the `InvOut` output parameter.

Use the Mode input parameter to define how the block should be used. Refer to the TimerP functions (Page 2174) section for more on this.

#### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

#### Startup characteristics

The block does not have any startup characteristics.

A startup behavior can be reached with a connection of the input parameter `Init` to the output parameter `InitOut` of the Trigger block. For more information, refer to Startup characteristics over Trigger block (Page 75).

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

TimerP modes (Page 2174)
TimerP error handling (Page 2177)
TimerP messaging (Page 2178)
TimerP I/Os (Page 2178)
TimerP block diagram (Page 2180)

## 12.1.2 TimerP modes

### TimerP operating modes

This block does not have any modes.

### See also

Description of TimerP (Page 2173)
TimerP functions (Page 2174)
TimerP error handling (Page 2177)
TimerP messaging (Page 2178)
TimerP I/Os (Page 2178)
TimerP block diagram (Page 2180)

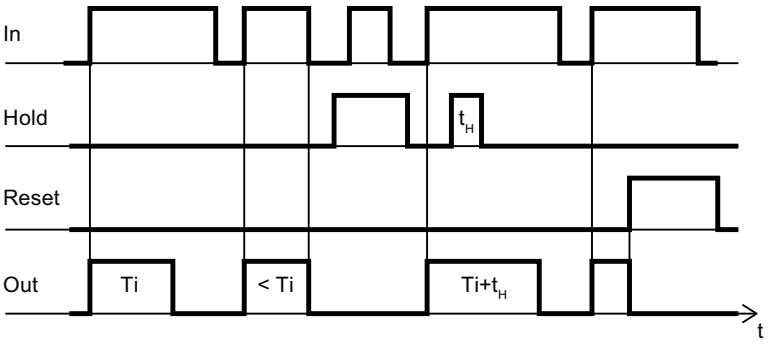
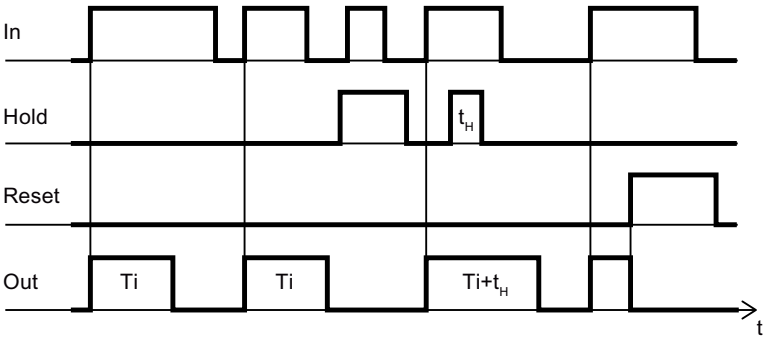
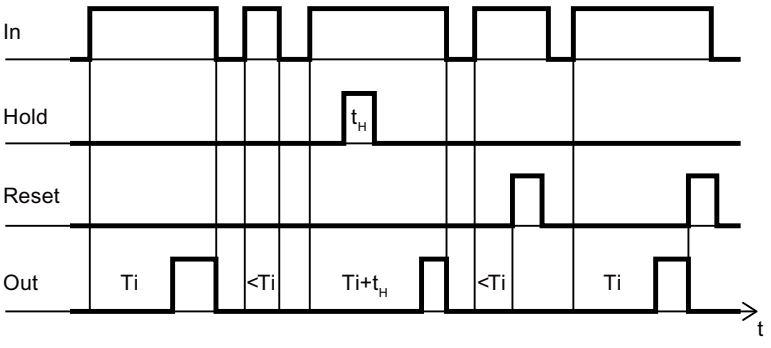
## 12.1.3 TimerP functions

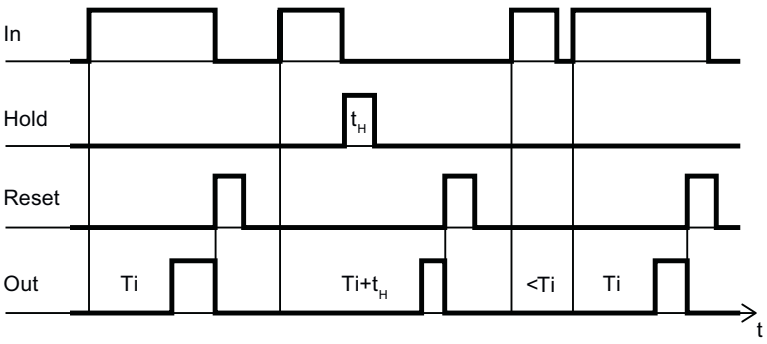
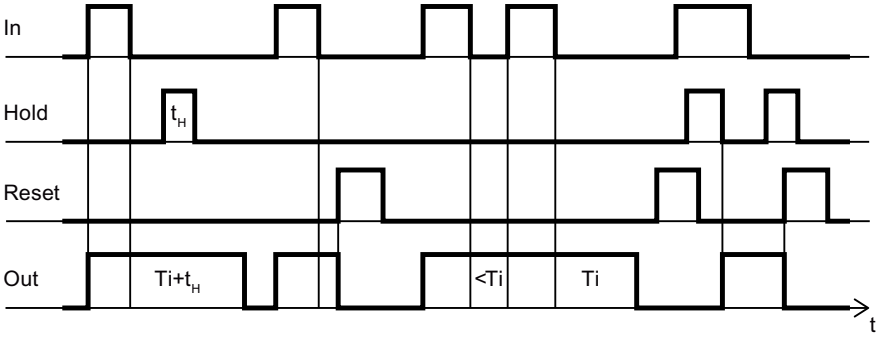
### Functions of TimerP

The functions for this block are listed below.

## Specifying how TimerP works

Use the `Mode` input parameter to specify how the block should work:

| Mode = | How the block works | Graphic representation |
|---|---|---|
| 0 | Start timer as pulse |  |
| 1 | Start extended pulse timer |  |
| 2 | Start ON delay timer |  |

| Mode = | How the block works | Graphic representation |
|---|---|---|
| 3 | Start ON delay timer with memory |  |
| 4 | Start OFF delay timer |  |

## Setting the time

You can use the `Ti` parameter to time for operating the block.

**Note**

Be aware when configuring this, that the time interval between `Ti` and `SampleTime` should not be more than $10^7$.

## Holding the `Out` and `TimeRemaining` output parameters

Use the `Hold.Value` = 1 parameter to freeze the calculation of the outputs `Out` and `TimeRemaining.Value`.

## Resetting the `Out` and `TimeRemaining` output parameters

Use the `Reset.Value` = 1 parameter to reset the output parameters `Out` and `TimeRemaining.Value` to 0.

## Initialization

With the input parameter `Init` = 1, the output parameters `Out` and `TimeRemaining` can be reset to 0 (same behavior as `Reset.Value` = 1). You can use the input parameter `Init` to realize a startup behavior.

To initialize the output parameters `Out` and `TimeRemaining` after a startup, connect the input parameter `Init` with the output parameter `InitOut` of the Trigger block. For more information, refer to Startup characteristics over Trigger block (Page 75).

### Forming the signal status for blocks

The signal status for the block is formed using the following parameters and output at the `Out` and `InvOut` output parameters:

- `In.ST`

### See also

Description of TimerP (Page 2173)

TimerP modes (Page 2174)

TimerP error handling (Page 2177)

TimerP messaging (Page 2178)

TimerP I/Os (Page 2178)

TimerP block diagram (Page 2180)

## 12.1.4 TimerP error handling

### Error handling of TimerP

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 1 | Invalid function type for the block set at the Mode input parameter |
| 11 | `Ti.Value` < `SampleTime`<br>This error causes the `TimeRemaining` and `Out` output parameters to be set to 0. |
| 33 | The value of `Ti` can no longer be displayed in the REAL number field. |

### See also

Description of TimerP (Page 2173)
TimerP modes (Page 2174)
TimerP functions (Page 2174)
TimerP messaging (Page 2178)
TimerP I/Os (Page 2178)
TimerP block diagram (Page 2180)

## 12.1.5 TimerP messaging

### Messaging

This block does not offer messaging.

### See also

Description of TimerP (Page 2173)
TimerP modes (Page 2174)
TimerP functions (Page 2174)
TimerP error handling (Page 2177)
TimerP I/Os (Page 2178)
TimerP block diagram (Page 2180)

## 12.1.6 TimerP I/Os

### I/Os of TimerP

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Hold | Hold calculation process | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| In | Input signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Init | Initialize function block | BOOL | 0 |
| Mode | How the block works:<br>0 = Start timer as pulse<br>1 = Start timer as extended pulse<br>2 = Start timer with ON delay<br>3 = Start timer with latching ON delay<br>4 = Start timer with OFF delay | INT | 2 |
| Reset | 1 = Reset output Out | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| Ti* | Time [s] | REAL | 0.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see TimerP error handling (Page 2177). | INT | -1 |
| InvOut | Inverted output signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| Out | Output signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| TimeRemaining | Remaining time [s] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

## See also

Description of TimerP (Page 2173)

TimerP modes (Page 2174)

TimerP functions (Page 2174)

TimerP messaging (Page 2178)
TimerP block diagram (Page 2180)

## 12.1.7 TimerP block diagram

### TimerP block diagram

A block diagram is not provided for this block.

### See also

Description of TimerP (Page 2173)
TimerP modes (Page 2174)
TimerP functions (Page 2174)
TimerP error handling (Page 2177)
TimerP messaging (Page 2178)
TimerP I/Os (Page 2178)

# 12.2 TimeTrig - Calculations with the date formats DT and TIME

## 12.2.1 Description of TimeTrig

**Object name (type + number) and family**

Type + number: FB 1802

Family: Time

### Area of application of TimeTrig

The block is used for the following applications:

- Minute, Daily, weekly or monthly generation of a pulse (trigger) using a definable time in minutes (0 to 59m), hours (0 to 23h) and day of the week (Sun to Sat) or one day (1 to 31) per month.
- Delayed generation a pulse (trigger) based on the current CPU time and a delay time in seconds (0 to 20 days).

### How it works

When a trigger is enabled, the next trigger time point and the duration until the next trigger time point is determined based on the CPU time. A periodic trigger and additionally a single trigger can be generated:

1. Generate periodic trigger (`PerTrigOn.Value = 1`)
2. Generate single trigger (`(SglTrigOn.Value = 1`)

The single trigger can be delayed by a specified period of time. The maximum delay is 20 days.

The period of time until the next trigger time point is displayed in days, hours, and minutes.

For calculations with the time formats DT and TIME, the block calls the following functions:

| Name | as of APL V8.1 | Family | No. in IEC | Symbol Comment |
|---|---|---|---|---|
| AD_DT_TM_APL | FC455 | IEC | FC1 | DT + add time |
| GE_DT_APL | FC456 | IEC | FC12 | Greater than or equal to DT |
| LE_DT_APL | FC452 | IEC | FC18 | Less than or equal to DT |
| LT_DT_APL | FC453 | IEC | FC23 | Smaller than DT |
| SB_DT_DT_APL | FC454 | IEC | FC34 | DT - DT subtract |

They come from the standard library "IEC function blocks" and are now also applied in the APL.

F1 will open the help from the original IEC function block.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (`OB30` to `OB38`). The block is first installed automatically in the startup OB (`OB100`).

### Startup characteristics

Define the startup characteristics for this block via `Feature.Bit0`:

- Bit 0: Set startup characteristics (Page 148)

### Status word allocation for `Status1` parameter

Refer to the following section for a description of the individual parameters: TimeTrig I/Os (Page 2192).

| Status bit | Parameter |
|---|---|
| 0 | Occupied |
| 1 | `BatchEn` |
| 2 | Not used |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | Next trigger is a Minute trigger |
| 6 | `OnAct.Value` |
| 7 | All triggers are switched off:<br>NOT `PerTrigOn.Value OR SglTrigOn.Value OR Trigger.Value` |
| 8 | `PerTrigOn.Value` |
| 9 | NOT `PerTrigOn.Value` |
| 10 | `SglTrigOn.Value` |
| 11 | NOT `SglTrigOn.Value` |
| 12 | `Trigger.Value` |
| 13 | Internal error |
| 14 | Next trigger is a daily trigger |
| 15 | Next trigger is a weekly trigger |
| 16 | Next trigger is a monthly trigger |
| 17 | Next trigger is a single trigger |
| 18 | `PerMode` = 1: Periodic trigger set to daily |
| 19 | `PerMode` = 2: Periodic trigger set to weekly |
| 20 | `PerMode` = 3: Periodic trigger set to monthly |
| 21 | Trigger for weekdays is Sunday |
| 22 | Trigger for weekdays is Monday |
| 23 | Trigger for weekdays is Tuesday |
| 24 | Trigger for weekdays is Wednesday |
| 25 | Trigger for weekdays is Thursday |
| 26 | Trigger for weekdays is Friday |
| 27 | Trigger for weekdays is Saturday |
| 28 | `LiOp.Value` |
| 29 | 1 = Invalid signal status<br>`SetPerLi.Value` and `RstPerLi.Value`,<br>`SetSglLi.Value` and `RstSglLi.Value` |

| Status bit | Parameter |
|---|---|
| 30 | `PerMode` = 4: Periodic trigger set to minute |
| 31 | Not used |

### See also

TimeTrig modes (Page 2183)

TimeTrig functions (Page 2184)

TimeTrig error handling (Page 2191)

TimeTrig messaging (Page 2192)

TimeTrig block diagram (Page 2197)

## 12.2.2 TimeTrig modes

### TimerTrig operating modes

The block can be operated using the following modes:

- On (Page 77)
- Out of service (Page 77)

### "On"

General information on the "On" mode is available in the section On (Page 77).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit5` =1 and `Feature.Bit5` =0 is as shown below:

| Output Parameters | **`Feature.Bit5` =0** | **`Feature.Bit5` =1** |
|---|---|---|
| PerTrigOn.ST | 16#XX | 16#60 |
| SglTrigOn.ST | 16#XX | 16#60 |
| Trigger.ST | 16#XX | 16#60 |

### See also

Description of TimeTrig (Page 2181)

TimeTrig functions (Page 2184)

TimeTrig error handling (Page 2191)

TimeTrig messaging (Page 2192)
TimeTrig I/Os (Page 2192)
TimeTrig block diagram (Page 2197)

## 12.2.3 TimeTrig functions

### Functions of TimeTrig

The Time Trig block reads the CPU time for every cycle, if the Local or External time is enabled, based on the enabled signal, local or external time is considered as the input time.

The functions for this block are listed below.

### Periodic output of a trigger

Use the `LiOp` input parameter to define whether the periodic trigger can be enabled (0 - 1, `SetPerOp` or `SetPerLi` parameter) or disabled (1 - 0, `RstPerOp` or `RstPerLi` parameter) via the faceplate or an interconnection.

`LiOp` = 0: Enable/disable via faceplate (`SetPerOp` or `RstPerOp`)

`LiOp` = 1: Enable/disable via interconnection (`SetPerLi` or `RstPerLi`)

You can select the faceplate from four periodic trigger settings:

- Periodic daily minutely output `PerMode` = 4
  You set the time in minutes with the MinuteOfHour parameter.
  For example: Period mode selected is “Minute” and “minute of hour” is set to 10 and current CPU time is 9-hour 11-minute, so next periodic trigger will happen at 10th hour 10th minute.
- Periodic daily output `PerMode` = 1
  You set the time in hours with the `HourOfDay` parameter.
- Periodic weekly output `PerMode` = 2
  You set the day of the week and time of day with the `HourOfDay` and `DayOfWeek` parameters.
- Periodic monthly output `PerMode` = 3
  You set the day of the month and time of day with the `HourOfDay` and `DayOfMonth` parameters. If the day of a month is greater than the last day, the last day is used.

### Delayed output of a single trigger

Use the `LiOp` input parameter to define whether the single trigger can be set (0 - 1, `SetPerOp` or `SetPerLi` parameter) or reset (1 - 0, `RstPerOp` or `RstPerLi` parameter) via the faceplate or an interconnection.

`LiOp` = 0: Set/reset via faceplate (`SetPerOp` or `RstPerOp`)

`LiOp` = 1: Set/reset via interconnection (`SetPerLi` or `RstPerLi`)

The output is delayed by the delay time `DelayTime` starting from the current CPU time. The currently effective delay time is displayed at the `DlyTmEff` output.

### Calculation of the next trigger time point

The next trigger time point is displayed at the `NxTrgDT`, `NxTrgYear`, `NxTrgMon`, `NxTrgDay`, `NxTrgHour`, `NxTrgMin` and `NxTrgSec` outputs . The next trigger time point is calculated from an enabled periodic trigger and an activated single trigger.

If no trigger is activated, the the minimum DT#1990-01-01-0:0:0.0 is output at the `NxTrgDT` output.

### Calculating the time period until the next trigger time point

The period of time until the next trigger time point based on the current CPU time is output at the `DiffTmDay`, `DiffTmHour`, `DiffTmMin` and `DiffTmSec` outputs.

If no trigger is activated, 0 is set at the outputs.

### Reading back the trigger states and the single trigger time point for a complete download

The `PerTrigOn`, `SglTrigOn` trigger states and the single trigger time point are written back to inputs:

- `InPerTrigOn := PerTrigOn`
- `InSglTrigOn := SglTrigOn`
- `InSglTrigDT := SglTrigDT`

If you want to use the reset for a complete download, you must read back the marked parameters in addition to the operated and monitored parameters before a complete download.

The values are written back to the outputs at startup and when `Feature.Bit0` = 1.

Also, in section Set startup characteristics (Page 148), see under "Set startup characteristics for the TimeTrig block".

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 1 | Reaction to the out of service mode (Page 188) |
| 5 | Changes signal status of outputs in OOS (Page 208) |
| 6 | External Input to enable or Disable (Page 211) |
| 7 | Switch over to internal time enable or Disable (Page 211) |
| 8 | AS time enable or Disable (Page 212) |
| 24 | Enabling local operator authorization (Page 169) |

### Operator permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the OS_Perm parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 = Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 = Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can enable the periodic trigger |
| 5 | 1 = Operator can disable the periodic trigger |
| 6 | 1 = Operator can set the single trigger |
| 7 | 1 = Operator can reset the single trigger |
| 8 | Not used |
| 9 | used in OSPermOut, OSPermLog for the display of the Enabled operations symbol of Periodic mode |
| 10 | 1 = Operator can enter HourOfDay |
| 11 | 1 = Operator can enter DayOfWeek |
| 12 | 1 = Operator can enter DayOfMonth |
| 13 | 1 = Operator can enter DelayDay |
| 14 | 1 = Operator can enter DelayHour |
| 15 | 1 = Operator can enter DelayMin |
| 16 | 1 = Operator can enter PulseWidth |
| 17 | 1 = Operator can set LocalTime |
| 18 | 1 = Operator can enter periodic mode Daily |
| 19 | 1 = Operator can enter periodic mode Weekly |
| 20 | 1 = Operator can enter periodic mode Monthly |
| 21 - 27 | used in OSPermLog for the weekdays Sun, Mon ... Sat |
| 28 | 1 = Operator can enter periodic mode Minute |
| 29 | 1 = Operator can enter MinuteOfHour |
| 30 - 31 | Not used |

## Recognizing daylight saving time

The block reads the identifier for daylight saving time from the CPU and writes it to the SummerTime output. The update occurs only when

- The MinuteOfHour, HourOfDay, DayOfWeek, DayOfMonth, DelayTime, PerMode = 1, PerMode = 2, PerMode = 3, PerMode = 4 or LocalTime parameters are changed
- A PerTrigOn or SglTrigOn trigger is enabled
- At startup or leaving Out of Service

If an error occurs when reading the daylight saving time, the status SummerTime.ST := 16#00 is set and the display is not updated. The status is not used in the calculation of ST_Worst.

### Local time

In every cycle, the block reads the CPU time. With `LocalTime` = 1 the block calculates a local time. To this a correction value will be read from CPU and build a difference time, which is added to the CPU time. This correction takes into account the time zone and the time difference in summer time (daylight savings time) and winter time (standard time).

The update of the correction value occurs only when:

- The `MinuteOfHour`, `HourOfDay`, `DayOfWeek`, `DayOfMonth`, `DelayTime`, `PerMode` = 1, `PerMode` = 2, `PerMode` = 3, `PerMode` = 4, or `LocalTime` parameters are changed.
- A `PerTrigOn` or `SglTrigOn` trigger is enabled.
- At startup or leaving "Out of Service" mode.

If an error occurs while reading the correction value, the status `SummerTime.ST` is set to 16#00 and the block does not changes the CPU time.

Local Time:If the LocalTime input is enabled, the block considers the local time as the current time irrespective of Feature Bit status.

If the Current time signal is Local AS time, then the output signal DstCurr is set to one.

When the Local time input is enabled then the output DstLocalAct is set to one.

In addition to feature bits, in order to consider the local time, Local time should be enabled in the CPU properties of the PLC.

The output time is obtained as below:

`Local_DT` = Local Date and Time
`Local_Year` = Local Time year
`Local_Mon` = Local Time Month
`Local_Day` = Local Time Day
`Local_Hou` = Local Time Hour
`Local_Min` = Local Time Minute
`Local_Sec` = Local Time Second

`CPU_DT` = Local Time Date and Time
`CPU_Year` = Local Time year
`CPU_Mon` = Local Time Month
`CPU_Day` = Local Day
`CPU_Hou` = Local Time Hour
`CPU_Min` = Local Time Minute
`CPU_Sec` = Local Time Second

### External Time

When the external time (ExtTime) is given, block uses the external time as the input time and the related outputs (CPU_DT = external time) is set. For the external time setting, for example, the block `BT_LT` (FC 61) can be used.

If the `Feature.Bit6` (=1) is active, and if the `Feature.Bit7` (=0) is not enabled, then external time is considered as the Current time for the generation of trigger pulses.

The output time is obtained as below:

`Ext_DT w` = ExtTime Date and Time
`Ext_Year w` = ExtTime year

Ext_Mon w = ExtTime Month
Ext_Day w = ExtTime Day
Ext_Hou w = ExtTime Hour
Ext_Min w = ExtTime Minute
Ext_Sec w = ExtTime Second

CPU_DT = ExtTime Date and Time
CPU_Year = ExtTime year
CPU_Mon = ExtTime Month
CPU_Day = ExtTime Day
CPU_Hou = ExtTime Hour
CPU_Min = ExtTime Minute
CPU_Sec = ExtTime Second

If the "External day light saving time signal" (DstExtTime) is enabled, then the output DstExtAct is set to one.

If the Current time signal is External time and if the day light saving time (DstExtTime) is enabled for the external time, then the output signal DstCurr is set to one.

### AS Time

When AS time is considered then the output (CPU_DT = AS time) is set.

- If the Feature.Bit8=1, Feature.Bit7=1 and if the external signal is of bad status then AS time is considered as Current time
- If the Feature.Bit6=0 and Feature.Bit8=1 then AS time is considered.

The output time is obtained as below:

AS_DT = AS Date and Time
AS_Year = AS Time year
AS_Mon = AS Time Month
AS_Day = AS Time Day
AS_Hou = AS Time Hour
AS_Min = AS Time Minute
AS_Sec = AS Time Second

CPU_DT = AS Time Date and Time
CPU_Year = AS Time year
CPU_Mon = AS Time Month
CPU_Day = ASTime Day
CPU_Hou = AS Time Hour
CPU_Min = AS Time Minute
CPU_Sec = AS Time Second

If the Current time signal is AS time, then the output signal DstCurr is set to zero.

### Automatic switchover of time signal

Feature.Bit7 acts as the automatic switch over of time signal and this feature is only effective if the 'External time signal(Feature Bit6)' is active, and if the Feature.Bit7 is enabled and the quality code of the external time signal is bad, the block uses the AS time as the basis for Time input.

Example for considering Local time input

- If the input LocalTime is set to one then the Local time is considered as the current time, irrespective of Feature Bits state.
- In the CPU Properties enable the local time.

| Feature Bits | Description | Type | Default |
|---|---|---|---|
| Bit6 | With this function you can activate external time.<br>• If 1 = external time is used for Switching.<br>• If 0 = external time cannot be used for switching | Bool | 0 |
| Bit7 | With this function you can choose internal time or AS Time.<br>• If 1 = switch to internal time(AS time) and if `ExtTime.ST`<>16#80 or 16#60<br>• If 0= internal time not used (AS time). | Bool | 0 |
| Bit8 | With this fuction you can choose AStime.<br>• If 1 = AS internal time<br>• If 0=AS time not used for Current time output | Bool | 0 |

| The probability of Inputs and Outputs for the time trig block | | | | | |
|---|---|---|---|---|---|
| **`FeatureBit6`** | **`FeatureBit7`** | **`FeatureBit8`** | **Local time Connected** | **External time Connected** | **Current Time considered** |
| External (external time used for switching) | Switch to internal time if the signal quality is bad | 1 = AS internal time | - | - | - |
| 0 | 0 | 0 | - | x | CPU TIME OR AS TIME |
| 0 | 0 | 1 | - | x | CPU TIME OR AS TIME |
| 0 | 1 | 0 | - | x | CPU TIME OR AS TIME |

| The probability of Inputs and Outputs for the time trig block | | | | | |
|---|---|---|---|---|---|
| **FeatureBit6** | **FeatureBit7** | **FeatureBit8** | **Local time Connected** | **External time Connected** | **Current Time considered** |
| 0 | 1 | 1 | - | x | CPU TIME OR AS TIME |
| 1 | 0 | 0 | - | 1 | EXTERNAL TIME |
| 1 | 0 | 1 | - | 1 | EXTERNAL TIME |
| 1 | 1 | 0 | - | 1 | BASED ON THE EXTERNAL SIGNAL STATUS, AS OR EXTERNAL TIME IS CONSIDERED |
| 1 | 1 | 1 | - | 1 | BASED ON THE EXTERNAL SIGNAL STATUS, AS OR EXTERNAL TIME IS CONSIDERED |
| x | x | x | 1 | 0 | AS LOCAL TIME=CPU + SUMMERTIME |
| x | x | x | x | 1 | BASED ON FEATURE BITS STATE, THE OUTPUT IS SET |
| NOTE: X represents 0 or 1 state | | | | | |

**Note**

External time is said to be connected if it is not equal to the default value.

Where default value is DT#2011-04-04-00:00:00.000

## Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The worst signal status `ST_Worst` for the block is formed from the following parameter:

- `SetSglLi.ST`
- `RstSglLi.ST`
- `SetPerLi.ST`

- RstPerLi.ST
- ExtTimeQc

The worst signal status ST_Worst is written to the following outputs:

- SglTrigOn.ST
- PerTrigOn.ST
- Trigger.ST

If an error occurs when reading the daylight saving time, the status SummerTime.ST := 16#00 is set and the display is not updated. The status is not used in the calculation of ST_Worst.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### See also

Description of TimeTrig (Page 2181)

TimeTrig modes (Page 2183)

TimeTrig error handling (Page 2191)

TimeTrig messaging (Page 2192)

TimeTrig I/Os (Page 2192)

TimeTrig block diagram (Page 2197)

## 12.2.4 TimeTrig error handling

### Error handling for TimeTrig

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

### Overview of error numbers

The ErrorNum I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed. |
| 0 | There is no error. |
| 51 | Invalid signal at LiOp =1<br>• SetPerLi = 1 and RstPerLi = 1<br>• SetSglLi = 1 and RstSglLi = 1 |

**See also**

Description of TimeTrig (Page 2181)
TimeTrig modes (Page 2183)
TimeTrig functions (Page 2184)
TimeTrig I/Os (Page 2192)
TimeTrig block diagram (Page 2197)

## 12.2.5 TimeTrig messaging

**Messaging**

This block does not offer messaging.

**See also**

TimeTrig modes (Page 2183)
TimeTrig functions (Page 2184)
TimeTrig error handling (Page 2191)
TimeTrig I/Os (Page 2192)
TimeTrig block diagram (Page 2197)
Description of TimeTrig (Page 2181)

## 12.2.6 TimeTrig I/Os

I/Os of TimeTrig

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

## Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| DstExtTime | Day light saving time for External input<br>DstExtTime=0 DST time not considered for external input<br>DstExtTime=1 DST considered for external input | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ExtTime | Input for External time of the day | STRUCT<br>• Value: DT<br>• ST: BYTE | -<br>1990-01-01-0:0:0.0 |
| InPerTrigOn | Periodic trigger 'On' (stored in the input variable) | BOOL | 0 |
| InSglTrigOn | Single trigger 'On' (stored in the input variable) | BOOL | 0 |
| InSglTrigDT | Single trigger data and time (stored in the input variable) | DATE_AND_TIME | 1990-01-01-0:00:00 |
| LiOp | Switchover of operating mode between:<br>1 = Interconnection<br>0 = Operator | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SetPerOp | Set periodic trigger via operator | BOOL | 0 |
| RstPerOp | Reset periodic trigger via operator | BOOL | 0 |
| SetPerLi | Set periodic trigger via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstPerLi | Reset periodic trigger via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SetSglOp | Set single trigger via operator | BOOL | 0 |
| RstSglOp | Reset single trigger via operator | BOOL | 0 |
| SetSglLi | Set single trigger via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RstSglLi | Reset single trigger via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PerMode | Periodic trigger;<br>1: daily, 2: weekly; 3: monthly | INT | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| DayOfWeek | Setpoint weekday for periodic trigger [1..7] | INT | 1 |
| DayOfMonth | Setpoint day of month for periodic trigger [1..31] | INT | 1 |
| MinuteOfHour | Setpoint minute of hour for periodic trigger [0..59] | INT | 0 |
| HourOfDay | Setpoint hour of day for periodic trigger [0..23] | INT | 0 |
| DelayTime | Delay time [sec] | DINT | 0 |
| PulseWidth | Pulse width of trigger signal [s] | REAL | 3.0 |
| SampleTime | Sampling time [s] | REAL | 0.1 |
| LocalTime | 1 = Local time ON | BOOL | 0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosOp* | 1 = "Out of service", via operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | STRING[32] | '' |
| StepNo | Batch step number | DWORD | 16#00000000 |
| Occupied | Occupied by batch | BOOL | 0 |
| SelFp1 | Open faceplate 1 | ANY | |
| SelFp2 | Open faceplate 2 | ANY | |
| OS_Perm | I/O for operator permissions (Page 2184) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| OpSt_In | Input parameter for local operator authorization, to be connected with the Out output parameter of the upstream block, "OpStations" | DWORD | 16#00000000 |
| Feature | I/O for additional functions (Page 2184) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| AS_DT | AS(CPU Time) Date and Time | DT | 1990-01-01-0:0:0.0 |
| AS_Year | AS Time Year | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| AS_Mon | AS Time Month | INT | 0 |
| AS_Day | AS Time Day | INT | 0 |
| AS_Hour | AS Time Hour | INT | 0 |
| AS_Min | AS Time Minute | INT | 0 |
| AS_Sec | AS Time Second | INT | 0 |
| Ext_DTw | External Date and Time | DT | 1990-01-01-0:0:0.0 |
| Ext_Yearw | Ext Time Year | INT | 0 |
| Ext_Monw | Ext Time Month | INT | 0 |
| Ext_Dayw | Ext Time Day | INT | 0 |
| Ext_Hourw | Ext Time Hour | INT | 0 |
| Ext_Minw | Ext Time Minute | INT | 0 |
| Ext_Secw | Ext Time Second | INT | 0 |
| Local_DT | Local(Local CPU Time) Date and Time | DT | 1990-01-01-0:0:0.0 |
| Local_Year | Local Time Year | INT | 0 |
| Local_Mon | Local Time Month | INT | 0 |
| Local_Day | Local Time Day | INT | 0 |
| Local_Hour | Local Time Hour | INT | 0 |
| Local_Min | Local Time Minute | INT | 0 |
| Local_Sec | Local Time Second | INT | 0 |
| DstCurr | Current daylight saving time<br>• DstCurr =0 day light saving not enabled<br>• DstCurr =1 day light saving enabled | BOOL | 0 |
| DstExtact | DstExtact=0 day light saving not enabled for external time<br>DstExtact=1 day light saving enabled for external time | BOOL | 0 |
| DstLocalact | DstLocalact =0 Local time enabled<br>DstLocalact =1 Local time disabled | BOOL | 0 |
| PerTrigOn | 1 = Periodic trigger On | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SglTrigOn | 1 = Single trigger On | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Trigger | Trigger signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CPU_Year | CPU time, year | INT | 0 |
| CPU_Mon | CPU time, month | INT | 0 |
| CPU_Day | CPU time, day | INT | 0 |
| CPU_Hour | CPU time, hour | INT | 0 |
| CPU_Min | CPU time, minute | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| CPU_Sec | CPU time, minute | INT | 0 |
| CPU_DayOfWeek | CPU time, weekday | INT | 0 |
| SglTrigDT | Single trigger date and time | DATE_AND_TIME | 1990-01-01-0:00:00 |
| NxTrgDT | Next trigger time point date and time | DATE_AND_TIME | 1990-01-01-0:00:00 |
| NxTrgYear | Next trigger time point year | INT | 0 |
| NxTrgMon | Next trigger time point months | INT | 0 |
| NxTrgDay | Next trigger time point days | INT | 0 |
| NxTrgHour | Next trigger time point hours | INT | 0 |
| NxTrgMin | Next trigger time point minutes | INT | 0 |
| NxTrgSec | Next trigger time point seconds | | |
| DiffTmDay | Day:<br>Period of time starting from the current CPU time until the next trigger time point | INT | 0 |
| DiffTmHour | Hours:<br>Period of time starting from the current CPU time until the next trigger time point | INT | 0 |
| DiffTmMin | Minutes:<br>Period of time starting from the current CPU time until the next trigger time point | INT | 0 |
| DiffTmSec | Seconds:<br>Period of time starting from the current CPU time until the next trigger time point | INT | 0 |
| DlyTmEff | Current effective delay time [sec] | DINT | 0 |
| BarTmSec | Bar graph seconds | DINT | 0 |
| BarOpHiScale | Bar graph minutes | DINT | 60 |
| DayOpHiScale | Scale graph days | INT | 1 |
| HrsOpHiScale | Scale graph hours | INT | 0 |
| MinOpHiScale | Scale graph minutes | INT | 0 |
| SummerTime | Daylight saving time | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OnAct | 1 = Block is in service | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OS_PermOut | Operator permission: Output for OS | DWORD | 16#FFFFFFFF |
| OS_PermLog | Operator permission: Output for OS | DWORD | 16#FFFFFFFF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status1 | Status word (Page 2181) | DWORD | 16#00000000 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see TimeTrig error handling (Page 2191). | INT | -1 |

**See also**

TimeTrig modes (Page 2183)

TimeTrig messaging (Page 2192)

TimeTrig block diagram (Page 2197)

## 12.2.7 TimeTrig block diagram

### Block diagram of TimeTrig

A block diagram is not provided for this block.

**See also**

TimeTrig modes (Page 2183)

TimeTrig functions (Page 2184)

TimeTrig error handling (Page 2191)

TimeTrig messaging (Page 2192)

TimeTrig I/Os (Page 2192)

## 12.2.8 Operator control and monitoring

### 12.2.8.1 TimeTrig views

### Views of the TimeTrig block

The block TimeTrig provides the following views:

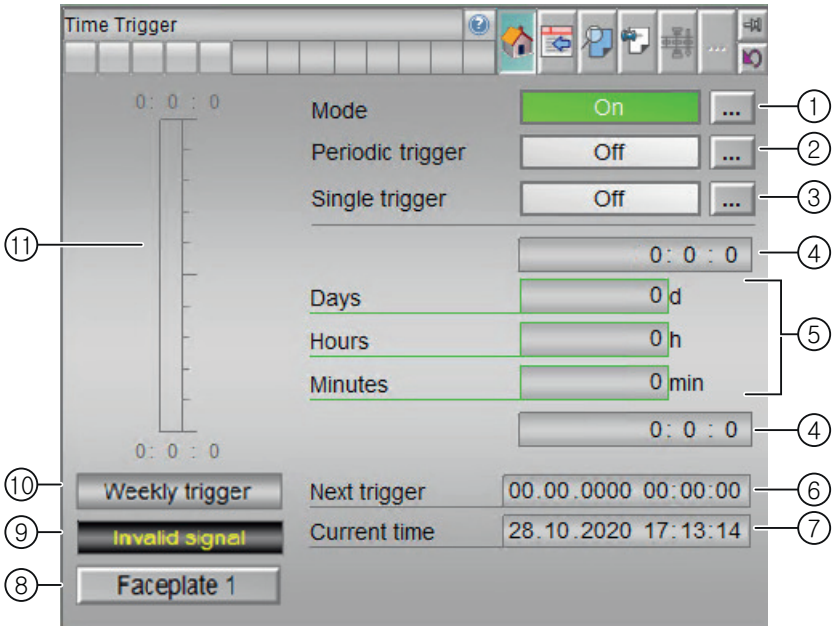
- TimeTrig standard view (Page 2198)
- TimeTrig parameter view (Page 2201)

- TimeTrig preview (Page 2203)
- Memo view (Page 338)
- Block icon for TimeTrig (Page 2205)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 12.2.8.2 TimeTrig standard view

#### Standard view of TimeTrig




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 77)
- Out of service (Page 77)

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

### (2) Enabling and disabling the period trigger

This area shows the default operating mode of the trigger. The following operating modes can be displayed and set here:

- "On"
- "Off"

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

### (3) Enabling and disabling the single trigger

This area shows the default operating mode of the trigger. The following operating modes can be displayed and set here:

- "On"
- "Off"

Refer to the Switching operating states and operating modes (Page 289) section for information on changing the state.

### (4) High and low scale range for the count value

This value provides information on the display range for the bar graph (above) of the count value. The scale range is defined in the engineering system.

### (5) Display of the count value

The following counts are shown here: The amount of time until the next trigger is shown here.

- "Days"
- "Hours"
- "Minutes"

### (6) Display of the time point of the next trigger

This area shows the time of the next trigger.

### (7) Displaying of the actual time of day

This area shows the actual time of day.

### (8) Navigation button for switching to the standard view of any faceplate

Use this navigation button to open the standard view of a block configured in the engineering system (ES). The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234) .

### (9) Display area for block states

This area provides additional information on the operating state of the block:

- "Internal error"
- "Invalid Signal"

### (10) Displaying of the actual trigger operating mode

- "Minute trigger"
- "Daily trigger"
- "Weekly trigger"
- "Monthly trigger"
- "Single trigger"

### (11) Graphic display of the current count value

This area shows you the current count in form of a bar graph. You can also see the high and low limits of the bargraph's scale. The visible area in the bar graph depends on the configuration in the engineering system (ES).

For more information, refer: Bar graph (Page 376)

### See also

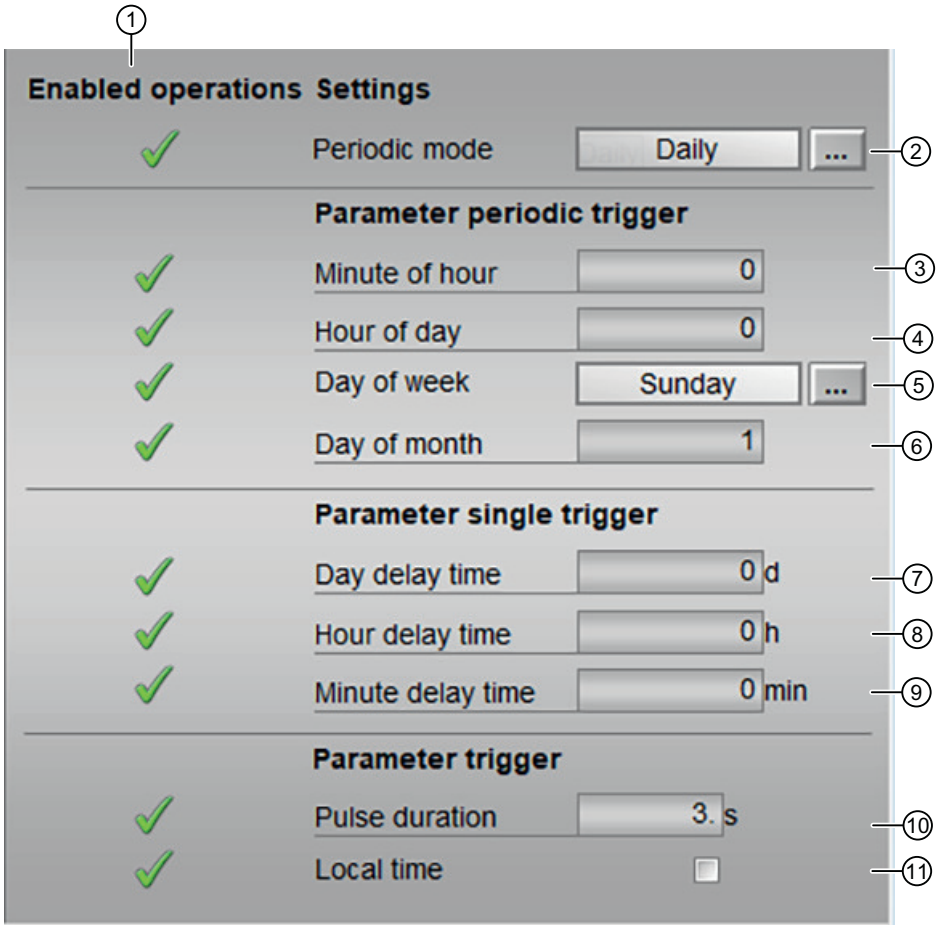
Release for maintenance (Page 70)

Simulating signals (Page 64)

Error handling (Page 128)

### 12.2.8.3 TimeTrig parameter view

**Parameter view of TimeTrig**




**(1) Enabled operation**

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`).

### (2) Periodic operating mode

You can change the operating mode of the periodic trigger:

- "Minute": The trigger is activated once an hour.
- "Daily": The trigger is activated once a day.
- "Weekly": The trigger is activated once a week.
- "Monthly": The trigger is activated once a month.

### (3), (4), (5) and (6) parameters for the periodic trigger

Enter the relevant parameters for the periodic mode:

- "Time": relevant for Minute, daily, weekly and monthly trigger
- "Weekday": relevant for weekly and monthly trigger
- "Day of the month": Relevant for monthly trigger. If the day of a month is greater than the last day, the last day is used.

### (7), (8) and (9) parameters for single triggers

Enter the parameters for the single trigger:

- "Delay time day"
- "Delay time hour"
- "Delay time minute"

### (10) and (11) parameters for the trigger

- "Pulse duration": Enter the duration of the trigger signal.
- "Local time": Activate local time by selecting the check box (☑).

### See also

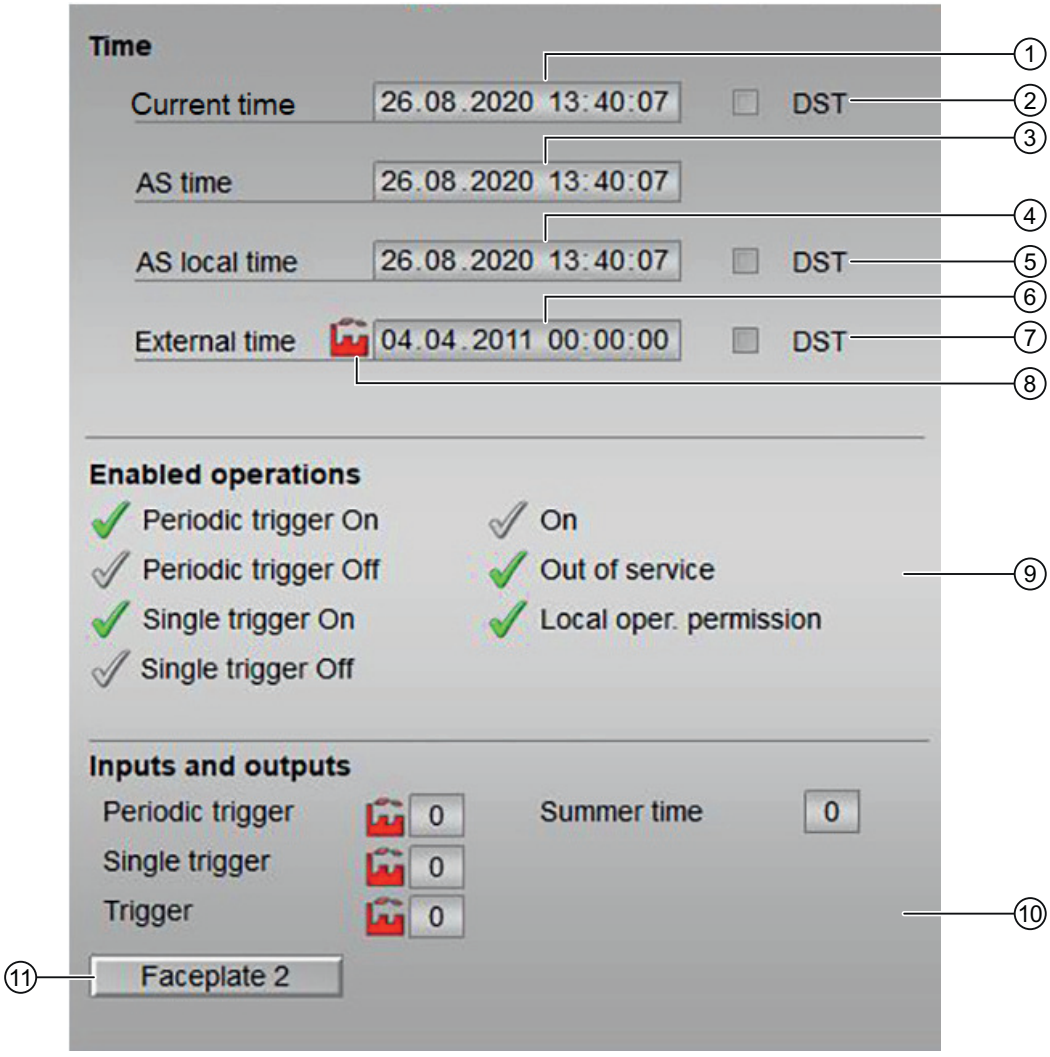
Changing values (Page 291)

Switching operating states and operating modes (Page 289)

Simulating signals (Page 64)

### 12.2.8.4 TimeTrig preview

**Preview of TimeTrig**




The screen shot also displays the Daylight Saving Time (DST) enabled or disabled status for the Current time, AS local time and for External time

**(1) This shows the current time which is active**

The current time may be AS, Local or External time, based on the parameter set through Feature Bits and Inputs.

**(2) Displays the DST**

Enabled or disabled status for the Current time.

### (3) Displays the CPU Time

### (4) Displays the AS local time

### (5) Displays the DST

Enabled or disabled status for the AS local time.

### (6) Displays the External time

### (7) Displays the DST

Enabled or disabled status for the External time.

### (8) External signal status

### (9) Enable operations

This area shows all operations for which special operator permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Icons for enabled operation:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- "Periodic trigger on": You can enable the periodic trigger.
- "Periodic trigger off": You can disable the periodic trigger.
- "Single trigger on": You can enable the single trigger.
- "Single trigger off": You can disable the single trigger.
- "On": You can operate the trigger.

- "Out of service": You can switch to "Out of service" operating mode.
- "Local operator permission": Use the ← button to switch to the standard view of the "OpStations" block.
  You can find additional information on this in the section Operator control permissions (Page 285).

### (10) Display of current operating signals

This area shows the most important parameters for this block with the current selection.

- "Periodic trigger":
  - 0 = Periodic trigger is enabled
  - 1 = Periodic trigger is disabled
- "Single trigger":
  - 0 = Single trigger is disabled
  - 1 = Single trigger is enabled
- "Trigger":
  - 0 = Output trigger signal is disabled
  - 1 = Output trigger signal is enabled
- "Daylight saving time":
  - 0 = Daylight saving time is disabled
  - 1 = Daylight saving time is enabled

### (11) Navigation button for switching to the standard view of any faceplate

Use this navigation button to open the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the section Opening additional faceplates (Page 234).

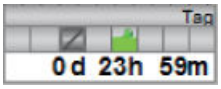
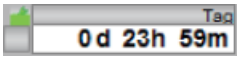
## 12.2.8.5 Block icon for TimeTrig

### Block icons for TimeTrig

A variety of block icons are available with the following functions:

- Process tag type
- Operating modes
- Signal status, release for maintenance
- Memo display
- "Time until the next trigger" display is running

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269).

# Mathematical blocks 13

## 13.1 AbsR - Absolute value of a real value

### 13.1.1 Description of AbsR

#### Object name (type + number) and family

Type + number: FC 395

Family: Math

#### Area of application for AbsR

The block is used for getting the absolute value of a real value.

#### How it works

The AbsR block provides the absolute value of a real value. The following equation is used for the calculation:

`Out` = |`In`|

Where:

`In` = Input value

`Out` = Absolute value

#### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

#### Startup characteristics

The block does not have any startup characteristics.

#### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

## 13.1.2 AbsR modes

### AbsR operating modes

This block does not have any modes.

## 13.1.3 AbsR functions

### Functions of AbsR

The functions for this block are listed below.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for mathematical blocks (Page 126).

The worst signal status for the block is formed by the following parameter and output at the `Out` output parameter:

- `In.ST`

## 13.1.4 AbsR error handling

### AbsR error handling

The block does not report any errors.

## 13.1.5 AbsR messaging

### Messaging

This block does not offer messaging.

## 13.1.6 AbsR I/Os

### AbsR I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In | Analog input value | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Analog output value | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |

## 13.1.7 AbsR block diagram

### AbsR block diagram

A block diagram is not provided for this block.

# 13.2 Add04 - Adder with 4 values

## 13.2.1 Description of Add04

### Object name (type + number) and family

Type + number: FC 351

Family: Math

### Area of application for Add04

The block is used for the following applications:

- Adding values
- Output of the total value for further processing

### How it works

The Add04 block calculates the sum of up to 4 values:

`Out` = `In1` +...+ `Inn` (`n` ≤ 4),

Where:

`Out` = sum value

`In1` ... `In4`= values to be added

The sum of all input parameters and the worst input parameter signal status is always output.

The output value is checked to determine if it is within the value range of REAL. If it is outside the value range, the highest or lowest possible REAL value is output.

If the value is not a number `(#NaN)`, the last valid output value is output and the status of the output value is set to 16#28 (if none of the input statuses is worse).

You can find additional information on forming the signal status under Forming and outputting the signal status for mathematical blocks (Page 126)

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

**See also**

Add04 functions (Page 2211)
Add04 messaging (Page 2212)
Add04 I/Os (Page 2213)
Add04 block diagram (Page 2214)
Add04 error handling (Page 2212)
Add04 modes (Page 2211)

## 13.2.2 Add04 modes

### Add04 operating modes

This block does not have any modes.

**See also**

Add04 block diagram (Page 2214)
Add04 I/Os (Page 2213)
Add04 messaging (Page 2212)
Add04 error handling (Page 2212)
Description of Add04 (Page 2210)
Add04 functions (Page 2211)

## 13.2.3 Add04 functions

### Functions of Add04

The functions for this block are listed below.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for mathematical blocks (Page 126).

The worst signal status for the block is formed by the following parameters and output at the `Out` output parameter:

- `In1.ST`
- `In2.ST`

- `In3.ST`
- `In4.ST`

**See also**

Description of Add04 (Page 2210)
Add04 messaging (Page 2212)
Add04 I/Os (Page 2213)
Add04 block diagram (Page 2214)
Add04 error handling (Page 2212)
Add04 modes (Page 2211)

## 13.2.4 Add04 error handling

**Add04 error handling**

The block does not report any errors.

**See also**

Add04 block diagram (Page 2214)
Add04 I/Os (Page 2213)
Add04 messaging (Page 2212)
Description of Add04 (Page 2210)
Add04 modes (Page 2211)
Add04 functions (Page 2211)

## 13.2.5 Add04 messaging

**Messaging**

This block does not offer messaging.

**See also**

Description of Add04 (Page 2210)
Add04 functions (Page 2211)
Add04 I/Os (Page 2213)

Add04 block diagram (Page 2214)
Add04 modes (Page 2211)
Add04 error handling (Page 2212)

## 13.2.6 Add04 I/Os

### Add04 I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In1 | Value 1 to add | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| In2 | Value 2 to add | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| In3 | Value 3 to add | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| In4 | Value 4 to add | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Output of the sum | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### See also

Description of Add04 (Page 2210)
Add04 functions (Page 2211)
Add04 messaging (Page 2212)

Add04 block diagram (Page 2214)
Add04 modes (Page 2211)
Add04 error handling (Page 2212)

## 13.2.7 Add04 block diagram

### Add04 block diagram

A block diagram is not provided for this block.

### See also

Description of Add04 (Page 2210)
Add04 modes (Page 2211)
Add04 functions (Page 2211)
Add04 error handling (Page 2212)
Add04 messaging (Page 2212)
Add04 I/Os (Page 2213)

# 13.3 Add08 - Adder with 8 values

## 13.3.1 Description of Add08

### Object name (type + number) and family

Type + number: FC 352

Family: Math

### Area of application for Add08

The block is used for the following applications:

- Adding values
- Output of the total value for further processing

### How it works

The Add08 block calculates the sum of up to 8 values:

`Out` = `In1` +...+ `Inn (n` ≤ 8),

Where:

`Out` = sum value

`In1` ... `In8` = values to be added

The sum of all input parameters and the worst input parameter signal status is always output.

The output value is checked to determine if it is within the value range of REAL. If it is outside the value range, the highest or lowest possible REAL value is output.

If the value is not a number `(#NaN)`, the last valid output value is output and the status of the output value is set to 16#28 (if none of the input statuses is worse).

You can find additional information on forming the signal status under Forming and outputting the signal status for mathematical blocks (Page 126)

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

Add08 functions (Page 2216)
Add08 messaging (Page 2217)
Add08 I/Os (Page 2218)
Add08 block diagram (Page 2219)
Add08 error handling (Page 2217)
Add08 modes (Page 2216)

## 13.3.2 Add08 modes

### Add08 operating modes

This block does not have any modes.

### See also

Description of Add08 (Page 2215)
Add08 functions (Page 2216)
Add08 error handling (Page 2217)
Add08 messaging (Page 2217)
Add08 I/Os (Page 2218)
Add08 block diagram (Page 2219)

## 13.3.3 Add08 functions

### Functions of Add08

The functions for this block are listed below.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for mathematical blocks (Page 126).

The worst signal status for the block is formed by the following parameters and output at the `Out` output parameter:

- `In1.ST`
- `In2.ST`
- `In3.ST`

- `In4.ST`
- `In5.ST`
- `In6.ST`
- `In7.ST`
- `In8.ST`

**See also**

Description of Add08 (Page 2215)
Add08 messaging (Page 2217)
Add08 I/Os (Page 2218)
Add08 block diagram (Page 2219)
Add08 error handling (Page 2217)
Add08 modes (Page 2216)

## 13.3.4 Add08 error handling

### Add08 error handling

The block does not report any errors.

**See also**

Add08 block diagram (Page 2219)
Add08 I/Os (Page 2218)
Add08 messaging (Page 2217)
Description of Add08 (Page 2215)
Add08 functions (Page 2216)
Add08 modes (Page 2216)

## 13.3.5 Add08 messaging

### Messaging

This block does not offer messaging.

### See also

Description of Add08 (Page 2215)
Add08 functions (Page 2216)
Add08 I/Os (Page 2218)
Add08 error handling (Page 2217)
Add08 modes (Page 2216)
Add08 block diagram (Page 2219)

## 13.3.6 Add08 I/Os

### Add08 I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In1 | Value 1 to add | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| In2 | Value 2 to add | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| In3 | Value 3 to add | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| In4 | Value 4 to add | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| In5 | Value 5 to add | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| In6 | Value 6 to add | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| In7 | Value 7 to add | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| In8 | Value 8 to add | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Output of the sum | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### See also

Description of Add08 (Page 2215)
Add08 functions (Page 2216)
Add08 messaging (Page 2217)
Add08 block diagram (Page 2219)
Add08 error handling (Page 2217)
Add08 modes (Page 2216)

## 13.3.7 Add08 block diagram

### Add08 block diagram

A block diagram is not provided for this block.

### See also

Description of Add08 (Page 2215)
Add08 functions (Page 2216)
Add08 error handling (Page 2217)
Add08 I/Os (Page 2218)
Add08 modes (Page 2216)
Add08 messaging (Page 2217)

# 13.4 Average - Mean value calculation

## 13.4.1 Description of Average

### Object name (type + number) and family

Type + number: FB1804

Family: Math

### Area of application for Average

The block is used for the following applications:

- Calculation of a time-based mean value

### How it works

The block calculates the time-based mean value of an analog value `In` based on the time which has expired since its start. The equation below is used:

$$Out = \frac{Out_{(n-1)} \cdot NumCycles + In}{NumCycles + 1}$$

Where:

`In` = input variable

`Out` = present average value

$Out_{(n-1)}$ = calculated average upon successful startup

`NumCycles` = Number of cycles for creating the mean value from the 0 - 1 edge transition of the Run input parameter.

The calculation is started by a 0 - 1 edge at the `Run` input parameter. The `Out` output parameter is overwritten by the `In` input parameter.

The result at output value `Out` is recalculated in the next cycles, and cycle counter `NumCycles` is incremented.

The `NumCycles` cycle counter has the DINT data type and can therefore assume the maximum value of 2147483647. When this value is reached, the `NumCycles` cycle counter is reset to 1. The formation of the mean is then started once again. At a cycle time of 100 ms, this case occurs every 6.8 years.

Calculation is terminated by resetting (1 - 0 edge) the `Run` input parameter. The last results set at `Out` and `NumCycles` are saved.

The parameter `Feature` can be used to specify the startup characteristics of the block.

You can find additional information on forming the signal status under Forming and outputting the signal status for mathematical blocks (Page 126)

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

### See also

Average functions (Page 2222)

Average messaging (Page 2223)

Average I/Os (Page 2224)

Average modes (Page 2221)

Average error handling (Page 2222)

Average block diagram (Page 2225)

## 13.4.2 Average modes

### Average operating modes

This block does not have any modes.

### See also

Description of Average (Page 2220)

Average functions (Page 2222)

Average error handling (Page 2222)

Average messaging (Page 2223)

Average I/Os (Page 2224)

Average block diagram (Page 2225)

## 13.4.3 Average functions

### Functions of Average

The functions for this block are listed below.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in section Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Functions |
|---|---|
| 0 | Set startup characteristics (Page 148) |

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for mathematical blocks (Page 126).

The signal status for the block is formed using the following parameters and output at the `Out` output parameter:

- `In.ST`

### See also

Description of Average (Page 2220)

Average messaging (Page 2223)

Average I/Os (Page 2224)

Average modes (Page 2221)

Average error handling (Page 2222)

Average block diagram (Page 2225)

## 13.4.4 Average error handling

### Error handling of Average

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Default value displayed after installation in the block. |
| 0 | There is no error. |
| 30 | The `In` value can no longer be displayed in the REAL number field, the 16#28 status is also output. The last valid value is provided at the `Out` output and calculation is halted. |
| 31 | The `Out` value can no longer be displayed in the REAL number field, the 16#28 status is also output.<br>The last valid value is provided at the `Out` output. |

### See also

Average modes (Page 2221)
Average functions (Page 2222)
Description of Average (Page 2220)
Average messaging (Page 2223)
Average I/Os (Page 2224)
Average block diagram (Page 2225)

## 13.4.5 Average messaging

### Messaging

This block does not offer messaging.

### See also

Description of Average (Page 2220)
Average functions (Page 2222)
Average I/Os (Page 2224)
Average modes (Page 2221)
Average error handling (Page 2222)
Average block diagram (Page 2225)

## 13.4.6 Average I/Os

### I/Os of Average

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 2222) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| In* | Analog input value for mean value calculation | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Run | 1 = Start time-based mean value calculation | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EndTime | 1 = Time for ending time-based mean value calculation | DT | 1990-01-01-0:00:00 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see Average error handling (Page 2222) | INT | -1 |
| NumCycles | Cycle counter | DINT | 1 |
| Out | Output for average value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| StartTime | 1 = Time for starting time-based mean value calculation | DT | 1990-01-01-0:00:00 |

**See also**

Description of Average (Page 2220)
Average messaging (Page 2223)
Average modes (Page 2221)
Average block diagram (Page 2225)

## 13.4.7 Average block diagram

**Average block diagram**

A block diagram is not provided for this block.

**See also**

Average modes (Page 2221)
Average functions (Page 2222)
Average error handling (Page 2222)
Average messaging (Page 2223)
Description of Average (Page 2220)
Average I/Os (Page 2224)

# 13.5 DeadTime - Delayed signal output

## 13.5.1 Description of DeadTime

### Object name (type + number) and family

Type + number: FB 1807

Family: Math

### Area of application for DeadTime

The block provides the following functions:

- Delay [s] of signal output

### How it works

This block delays the output of an input value by a time `DeadTime` [s] selected by the user.

Dead times up to 100 times the `SampleTime` can be realized. If you want to set a longer dead time, you need to connect several dead time blocks in sequence. You can also insert the dead time block in a runtime group with a longer sampling time or scale down the runtime group, but this would lead to loss of input values.

The block operates as follows:

`Out` = `In` (t - `DeadTime`)

Where:

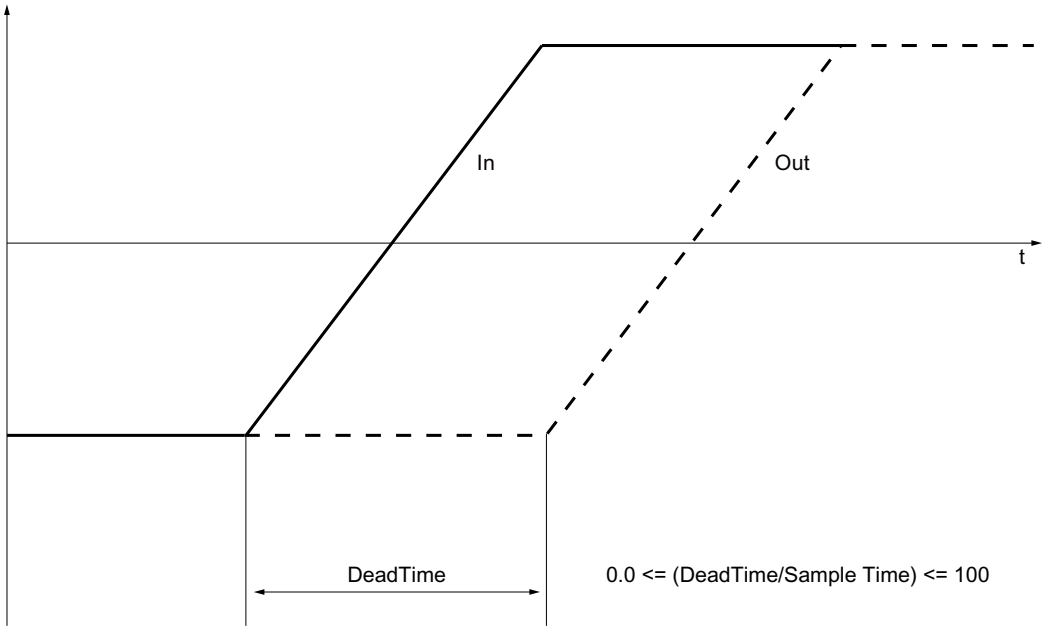
`Out` = output value

t = current time

`DeadTime` = `DeadT_Cyc` · `SampleTime` where:

`DeadT_Cyc` = Number of cycles [0...100], by which the analog input value is delayed

An analog value of input `In` is only output after the variable dead time at `DeadTime` has expired. It is output at `Out`.




You can find additional information on forming the signal status under Forming and outputting the signal status for mathematical blocks (Page 126)

**Note**

If you change the value for the dead time during the runtime of the block, the input value is written to the output. The change to the dead time then takes effect.

## Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

The resulting sampling time of the block must be long enough for the desired dead time to be specified at the `DeadTime` parameter.

For the DeadTime block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 2827)
- PID controller with Smith predictor (SmithPredictorControl) (Page 2830)

## Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

### See also

DeadTime functions (Page 2228)
DeadTime messaging (Page 2230)
DeadTime I/Os (Page 2230)
DeadTime block diagram (Page 2231)
DeadTime modes (Page 2228)
DeadTime error handling (Page 2229)

## 13.5.2 DeadTime modes

### DeadTime operating modes

This block does not have any modes.

### See also

DeadTime block diagram (Page 2231)
DeadTime I/Os (Page 2230)
DeadTime messaging (Page 2230)
DeadTime error handling (Page 2229)
DeadTime functions (Page 2228)
Description of DeadTime (Page 2226)

## 13.5.3 DeadTime functions

### Functions of DeadTime

The functions for this block are listed below.

### Configurable reactions using the `Feature` I/O

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) section. The following functionality is available for this block at the relevant bits:

| Bit | Functions |
|---|---|
| 0 | Set startup characteristics (Page 148) |

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for mathematical blocks (Page 126).

The signal status for the block is formed using the following parameters and output at the `Out` output parameter:

- `In.ST`

### See also

Description of DeadTime (Page 2226)

DeadTime messaging (Page 2230)

DeadTime I/Os (Page 2230)

DeadTime block diagram (Page 2231)

DeadTime modes (Page 2228)

DeadTime error handling (Page 2229)

## 13.5.4 DeadTime error handling

### Error handling of DeadTime

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; block is not processed. |
| 0 | There is no error. |

| Error number | Meaning of the error number |
|---|---|
| 30 | The value of `In` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out` output. |
| 32 | The value of `DeadTime` can no longer be displayed in the REAL number field.<br>The last valid value is provided at the `Out` output. |
| 45 | The value of `DeadTime` is outside the permitted range<br>`0` ≤ `INT` (`DeadTime` / `SampleTime`) ≤ 100<br>If the value of `DeadTime` is greater than 100 · `SampleTime`, the block works internally with the maximum dead time of 100 · `SampleTime`. |

**See also**

DeadTime block diagram (Page 2231)
DeadTime I/Os (Page 2230)
DeadTime messaging (Page 2230)
DeadTime modes (Page 2228)
Description of DeadTime (Page 2226)
DeadTime functions (Page 2228)

## 13.5.5 DeadTime messaging

### Messaging

This block does not offer messaging.

**See also**

Description of DeadTime (Page 2226)
DeadTime functions (Page 2228)
DeadTime I/Os (Page 2230)
DeadTime block diagram (Page 2231)
DeadTime modes (Page 2228)
DeadTime error handling (Page 2229)

## 13.5.6 DeadTime I/Os

### I/Os of DeadTime

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| DeadTime* | Dead time [s]: Time by which the analog input value is delayed | REAL | 1.0 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 2228) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| In* | Analog input value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see DeadTime error handling (Page 2229). | INT | -1 |
| DeadT_Cyc | Number of cycles [0 to 100] by which the analog input value is delayed | INT | 0 |
| Out | Analog output value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### See also

Description of DeadTime (Page 2226)

DeadTime messaging (Page 2230)

DeadTime block diagram (Page 2231)

DeadTime modes (Page 2228)

## 13.5.7 DeadTime block diagram

### DeadTime block diagram

A block diagram is not provided for this block.

**See also**

DeadTime I/Os (Page 2230)
DeadTime messaging (Page 2230)
DeadTime error handling (Page 2229)
DeadTime functions (Page 2228)
DeadTime modes (Page 2228)
Description of DeadTime (Page 2226)

# 13.6 Derivative - Obtaining a derivative

## 13.6.1 Description of Derivative

### Object name (type + number) and family

Type + number: FB 1808

Family: Math

### Area of application for Derivative

The block is used for the following applications:

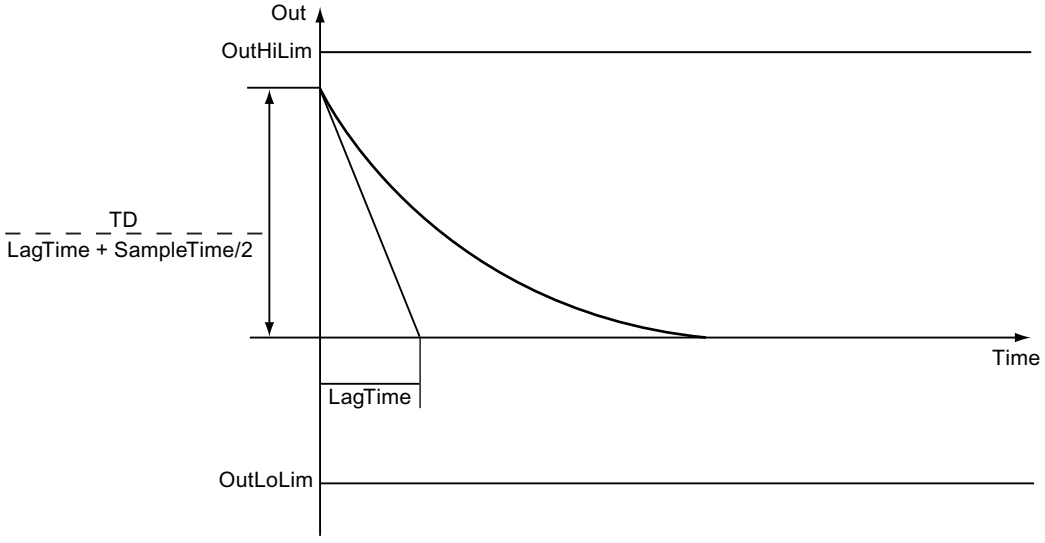
- Forming the D component for the controller design
- Forming a temporal derivative of the input signal

### How it works

The block can be used as part of user-designed closed-loop controller. The block calculates a D component with delay. The derivative of the input signal is formed using the "trapezoid rule".

If only the derivative signal is to be calculated, `TD` = 1 [s] must be set. The speed signal is then formed from the travel signal.

The output signal can be damped using the `LagTime` parameter. Damping is disabled by configuring `LagTime` = 0. The following figure shows the step response from the unit step of the `In` input variable.




The block works according to the following formula:

$$Out_{(n)} = \left( \frac{TD}{LagTime + \frac{SampleTime}{2}} \right) \left( In_{(n)} - PrevIn_{(n)} \right)$$

The $PrevIn_{(n)}$ value is calculated for the next sampling step by means of a first order low pass filter:

$$PrevIn_{(n+1)} = PrevIn_{(n)} + \frac{SampleTime}{TD} Out_{(n)}$$

The following applies for both formulas:

- `Out`$_{(n)}$ = output value
- `TD` = derivative time
- `In`$_{(n)}$ = input value
- `PrevIn`$_{(n)}$ = memory variable of the low pass filter
- `SampleTime` = sampling time [s]
- `LagTime` = delay time [s]

You can find additional information on forming the signal status under Forming and outputting the signal status for mathematical blocks (Page 126)

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB3x). The block is then installed automatically in startup OB (OB100).

Further addressing is not required.

For the Derivative block, the Advanced Process Library contains a template for process tag types as an example with an application scenario for this block.

Example of process tag types:

- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 2827)

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

Derivative messaging (Page 2237)

Derivative error handling (Page 2237)

Derivative modes (Page 2235)

Derivative I/Os (Page 2238)

Derivative functions (Page 2236)

## 13.6.2 Derivative modes

### Derivative operating modes

This block does not have any modes.

### See also

Derivative block diagram (Page 2239)

Derivative I/Os (Page 2238)

Derivative messaging (Page 2237)

Derivative error handling (Page 2237)

Derivative functions (Page 2236)

Description of Derivative (Page 2233)

## 13.6.3 Derivative functions

### Functions of Derivative

The functions for this block are listed below.

### Monitoring the output parameter for limits

The `Out` output parameter can be checked for limit values:

- `OutHiLim`: High limit
- `OutLoLim`: Low limit

If the limits are infringed, this is indicated to you at the corresponding output parameters (output parameter `OutHiAct` or `OutLoAct` = 1).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in section Configurable functions using the Feature I/O (Page 139). The following behavior patterns are available for this block at the relevant bits

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for mathematical blocks (Page 126).

The signal status for the block is formed using the following parameters and output at the `Out`, `OutHiAct` and `OutLoAct` output parameters:

- `In.ST`

### See also

Description of Derivative (Page 2233)

Derivative messaging (Page 2237)

Derivative I/Os (Page 2238)

Derivative block diagram (Page 2239)

Derivative error handling (Page 2237)

Derivative modes (Page 2235)

## 13.6.4 Derivative error handling

### Error handling of Derivative

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; block is not processed |
| 0 | There is no error. |
| 30 | The value of `In` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out` output. |
| 31 | The value of `Out` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out` output. |
| 32 | The value of `LagTime` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out` output. |
| 33 | The value of `TD` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out` output. |

### See also

Derivative block diagram (Page 2239)

Derivative I/Os (Page 2238)

Derivative messaging (Page 2237)

Description of Derivative (Page 2233)

Derivative modes (Page 2235)

Derivative functions (Page 2236)

## 13.6.5 Derivative messaging

### Messaging

This block does not offer messaging.

### See also

Description of Derivative (Page 2233)
Derivative functions (Page 2236)
Derivative I/Os (Page 2238)
Derivative block diagram (Page 2239)
Derivative modes (Page 2235)
Derivative error handling (Page 2237)

## 13.6.6 Derivative I/Os

### I/Os of Derivative

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 2236) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| In* | Analog input value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| LagTime* | Delay time [s] | REAL | 10.0 |
| OutHiLim | Limit (high) for output value | REAL | 100.0 |
| OutLoLim | Limit (low) for output value | REAL | 0.0 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| TD* | Derivative time [s] | REAL | 1.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see Derivative error handling (Page 2237) | INT | -1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Out | Output value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| OutHiAct | 1= Limit (high) violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OutLoAct | 1= Limit (low) violated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### See also

Description of Derivative (Page 2233)
Derivative messaging (Page 2237)
Derivative block diagram (Page 2239)
Derivative modes (Page 2235)

## 13.6.7 Derivative block diagram

### Derivative block diagram

A block diagram is not provided for this block.

### See also

Description of Derivative (Page 2233)
Derivative modes (Page 2235)
Derivative functions (Page 2236)
Derivative error handling (Page 2237)
Derivative messaging (Page 2237)
Derivative I/Os (Page 2238)

# 13.7 Div02 - Division of two values

## 13.7.1 Description of Div02

### Object name (type + number) and family

Type + number: FC 358

Family: Math

### Area of application for Div02

The block is used for the following applications:

- Dividing two values
- Output of the division for further processing

### How it works

The block is used to divide two values as follows:

`Out` = `In1` / `In2`

If the input parameter is `In2` = 0, the last valid output value is retained until another division is permitted mathematically.

The worst signal status is also always output at the output parameter.

The output value is checked to determine if it is within the value range of REAL. If it is outside the value range, the highest or lowest possible REAL value is output.

If the value is not a number `(#NaN)` or divided by 0, the last valid output value is output and the status of the output value is set to 16#28 (if none of the input statuses is worse).

You can find additional information on forming the signal status under Forming and outputting the signal status for mathematical blocks (Page 126)

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

### See also

Div02 block diagram (Page 2244)
Div02 I/Os (Page 2243)
Div02 messaging (Page 2243)
Div02 error handling (Page 2242)
Div02 functions (Page 2241)
Div02 modes (Page 2241)

## 13.7.2 Div02 modes

### Div02 operating modes

This block does not have any modes.

### See also

Div02 block diagram (Page 2244)
Div02 I/Os (Page 2243)
Div02 messaging (Page 2243)
Div02 error handling (Page 2242)
Div02 functions (Page 2241)
Description of Div02 (Page 2240)

## 13.7.3 Div02 functions

### Functions of Div02

The functions for this block are listed below.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for mathematical blocks (Page 126).

The worst signal status for the block is formed by the following parameters and output at the `Out` output parameter:

- `In1.ST`
- `In2.ST`

### See also

Div02 block diagram (Page 2244)

Div02 I/Os (Page 2243)

Div02 messaging (Page 2243)

Div02 error handling (Page 2242)

Div02 modes (Page 2241)

Description of Div02 (Page 2240)

## 13.7.4 Div02 error handling

### Error handling of Div02

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; block is not processed |
| 0 | There is no error. |
| 30 | It has been divided by 0. |

### See also

Div02 block diagram (Page 2244)

Div02 messaging (Page 2243)

Div02 functions (Page 2241)

Div02 modes (Page 2241)

Description of Div02 (Page 2240)

Div02 I/Os (Page 2243)

## 13.7.5 Div02 messaging

### Messaging

This block does not offer messaging.

### See also

Div02 block diagram (Page 2244)
Div02 I/Os (Page 2243)
Description of Div02 (Page 2240)
Div02 modes (Page 2241)
Div02 functions (Page 2241)
Div02 error handling (Page 2242)

## 13.7.6 Div02 I/Os

### I/Os of Div02

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In1 | Dividend | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| In2 | Divisor | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see Div02 error handling (Page 2242). | INT | -1 |
| Out | Output of division | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### See also

Div02 block diagram (Page 2244)

Description of Div02 (Page 2240)

Div02 modes (Page 2241)

Div02 functions (Page 2241)

Div02 messaging (Page 2243)

## 13.7.7 Div02 block diagram

### Div02 block diagram

A block diagram is not provided for this block.

### See also

Div02 I/Os (Page 2243)

Div02 messaging (Page 2243)

Div02 error handling (Page 2242)

Div02 functions (Page 2241)

Div02 modes (Page 2241)

Description of Div02 (Page 2240)

# 13.8 FlowCorr - Flow correction

## 13.8.1 Description of FlowCorr

### Object name (type + number) and family

Type + number: FB 1916

Family: Math

### Area of application for FlowCorr

The block is used for the following applications:

- Pressure and/or temperature compensation for measurements of the differential pressure method

The block calculates the pressure and/or temperature compensation for measurements with the differential pressure method. This refers to measurements from restrictors, standard flow nozzles, venturi nozzles and pitot tubes. The block can be used for gases and vapors, saturated steam and liquids to calculate the volume flow rate or, for liquids, to calculate the mass flow rate.

### How it works

The block cyclically calculates the volume/mass flow rate.

Mode 1: Gas and steam:

In mode 1, the flow is calculated as shown in the following equation:

$$Q_{cal} = Q_0 * \sqrt{\frac{\Delta P}{\Delta P_0} * \frac{(P + P_B)}{(P_0 + P_B)} * \frac{(T_0 + T_{abs})}{(T + T_{abs})}}$$

Mode 2: Saturated steam, mass flow in liquids:

In mode 2, the flow is calculated as shown in the following equation:

$$Q_{cal} = Q_0 * \sqrt{\frac{\Delta P}{\Delta P_0} * \frac{\rho}{\rho_0}}$$

Mode 3: Volume flow for liquids:

In mode 3, the flow is calculated as shown in the following equation:

$$Q_{cal} = Q_0 * \sqrt{\frac{\Delta P}{\Delta P_0} * \frac{\rho_0}{\rho}}$$

| Block I/O | Icon | Type | Description |
|---|---|---|---|
| Flow | $Q_{cal}$ | Result | Calculated flow |
| Flow0 | $Q_0$ | Parameter | Design flow |
| DiffP | $\Delta P$ | Measurement | Differential pressure [1] |
| DiffP0 | $\Delta P_0$ | Parameter | Design differential pressure |
| P | $P$ | Parameter or measurement | Pressure [bar (abs. or relative)] [2] |
| P0 | $P_0$ | Parameter | Design pressure [bar (abs. or relative)] [3] |
| P_Atmos | $P_B$ | Parameter or measurement | Barometric pressure [mbar (abs.)] [4] |
| Temp0 | $T_0$ | Parameter | Design temperature [0 °C] |
| Temp | $T$ | Parameter or measurement | Temperature [0 °C] |
| TempAbs | $T_{abs}$ | Parameter | [0 °C] in Kelvin (corresponds to 273.15K) |
| Density | $\rho$ | Parameter or measurement | Density [5] |
| Density0 | $\rho_0$ | Parameter | Design density |

1 When the sensor extracts the square root of the differential pressure, the P_SqrtOn parameter must be set to 'true'. In this case, the differential pressure value is squared before the flow has been calculated by the above equation.

2 If a sensor is used for absolute pressure, P_Atmos must be set to '0'. If a sensor is used for differential pressure, P_Atmos must either be set to the local average ambient pressure or linked to the ambient pressure measurement.

3 'P0' must be the same size as 'P' (e.g.: if a sensor for absolute pressure is used for 'P', there must be a matching value for the absolute pressure 'P0').

4 Unit of P_Atmos = unit of P / 1000 (e.g.: P: bar; P_Atmos: mbar)

5 Real density measurement can be connected. An alternative method is often used in practice to determine the density (a process pressure measurement for saturated steam or a temperature measurement for liquids).

## Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

FlowCorr block diagram (Page 2252)
FlowCorr I/Os (Page 2250)
FlowCorr messaging (Page 2250)
FlowCorr error handling (Page 2249)
FlowCorr functions (Page 2247)
FlowCorr modes (Page 2247)
Forming and outputting the signal status for mathematical blocks (Page 126)

## 13.8.2 FlowCorr modes

### FlowCorr operating modes

This block does not have any operating modes.

### See also

FlowCorr block diagram (Page 2252)
FlowCorr I/Os (Page 2250)
FlowCorr messaging (Page 2250)
FlowCorr error handling (Page 2249)
FlowCorr functions (Page 2247)
Description of FlowCorr (Page 2245)

## 13.8.3 FlowCorr functions

### Functions of FlowCorr

The functions for this block are listed below.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for mathematical blocks (Page 126).

The worst signal status for the block is formed from the following parameters and output at the `Flow.ST` output parameter:

- `DiffP.ST`
- `DiffP0.ST`
- `P.ST`
- `P0.ST`
- `P_Atmos.ST`
- `Temp.ST`
- `Temp0.ST`
- `Density.ST`
- `Density0.ST`
- `Flow0.ST`
- `TempAbs.ST`

When one of the following conditions applies and depending on `Feature.Bit8` (Switch to substitute value (Page 159)), the last valid value is retained at the `Flow` output or the substitute value `SubsFlow` is applied:

- One of the inputs relevant to the calculation or the `Flow` output is outside the `REAL` counter range
- A division by zero occurs in the calculation
- The Mode parameter is not 1, 2 or 3

`Feature.Bit8` =0: Last valid value is retained and the signal status is set to bad, process-related `16#28` (assuming the signal status is not `16#0`).

`Feature.Bit8` =1: Substitute value `SubsFlow` is applied and the signal status is set to manipulated value (for example, substitute value, simulation, last valid value) `16#60`.

### See also

FlowCorr block diagram (Page 2252)

FlowCorr I/Os (Page 2250)

FlowCorr messaging (Page 2250)

FlowCorr error handling (Page 2249)

FlowCorr modes (Page 2247)

Description of FlowCorr (Page 2245)

## 13.8.4 FlowCorr error handling

### Error handling of FlowCorr

The block does not report any errors.

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

When one of the following conditions applies and depending on `Feature.Bit8` (Switch to substitute value (Page 159)) at the `Flow` output, the last valid value is retained or the substitute value `SubsFlow` is applied:

- One of the inputs relevant to the calculation or the `Flow` output is outside the `REAL` counter range
- A division by zero occurs in the calculation
- The Mode parameter is not 1, 2 or 3

`Feature.Bit8` = 0: Last valid value is retained and the signal status is set to bad, process-related `16#28` (assuming the signal status is not `16#0`).

`Feature.Bit8` = 1: Substitute value `SubsFlow` is applied and the signal status is set to manipulated value (for example, substitute value, simulation, last valid value) `16#60`.

The `Bad.Value` output is set to `true` when the following applies:

- When the signal status is formed, the value is bad, device-related or process-related (`16#00`, `16#28`)
- One of the inputs relevant to the calculation or the `Flow` output is outside the `REAL` counter range
- A division by zero occurs in the calculation
- The Mode parameter is not 1, 2 or 3

### Overview of error numbers

The block has no `ErrorNum` I/O.

### See also

FlowCorr block diagram (Page 2252)

FlowCorr I/Os (Page 2250)

FlowCorr messaging (Page 2250)

FlowCorr functions (Page 2247)

FlowCorr modes (Page 2247)

Description of FlowCorr (Page 2245)

## 13.8.5 FlowCorr messaging

### Messaging

This block does not offer messaging.

### See also

FlowCorr block diagram (Page 2252)
FlowCorr I/Os (Page 2250)
FlowCorr error handling (Page 2249)
FlowCorr functions (Page 2247)
FlowCorr modes (Page 2247)
Description of FlowCorr (Page 2245)

## 13.8.6 FlowCorr I/Os

### I/Os of FlowCorr

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `Density` | Density | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `Density0` | Design density | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `DiffP` | Differential pressure [bar] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |
| `DiffP0` | Design differential pressure [bar] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 100.0<br>• 16#80 |
| `Feature` | I/O for additional functions (Page 2247) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Flow0 | Design flow | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Mode | Mode 1..3 | INT | 1 |
| P_Atmos | Barometric pressure [mbar (abs)] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1000.0<br>• 16#80 |
| P_SqrtOn | 1 = Sensor calculates the square root | BOOL | 0 |
| P | Pressure [bar (abs or relative)] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| P0 | Design pressure [bar (abs or relative)] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |
| SubsFlow | Substitute value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Temp | Temperature [°C] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Temp0 | Design temperature [°C] | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 20.0<br>• 16#80 |
| TempAbs | Absolute temperature [K] at 0°C (corresponds to 273.15K) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 273.15<br>• 16#80 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Bad | 1 = Process value is not valid | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Flow | Calculated flow | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

## See also

FlowCorr block diagram (Page 2252)

FlowCorr messaging (Page 2250)

FlowCorr error handling (Page 2249)
FlowCorr modes (Page 2247)
Description of FlowCorr (Page 2245)

## 13.8.7 FlowCorr block diagram

### FlowCorr block diagram

A block diagram is not provided for this block.

### See also

FlowCorr I/Os (Page 2250)
FlowCorr messaging (Page 2250)
FlowCorr error handling (Page 2249)
FlowCorr functions (Page 2247)
FlowCorr modes (Page 2247)
Description of FlowCorr (Page 2245)

# 13.9 Integral - Generating a time integral

## 13.9.1 Description of Integral

### Object name (type + number) and family

Type + number: FB 1823

Family: Math

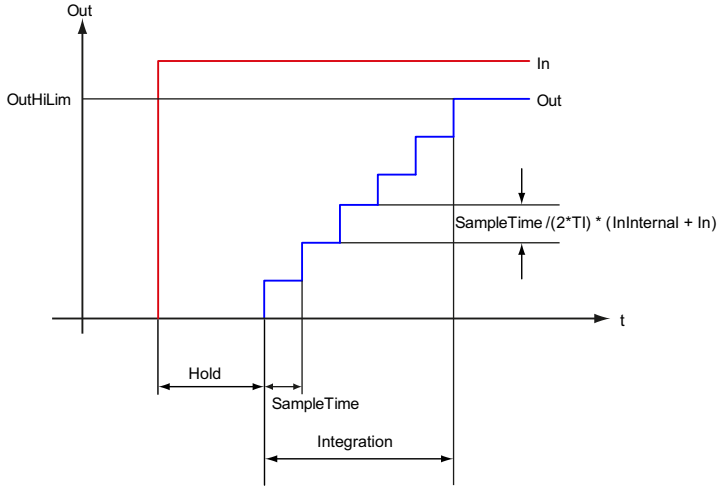
### Area of application for Integral

The block is used for the following application:

- Forming the time integral of the connected input signal
- Forming the D component for the controller design

### How it works

The block can be used as part of user-designed closed-loop controller. It integrates the input signal `In` based on the trapezoid rule and outputs the result, the integral action component, at `Out`. If the block should calculate the pure time integral, `TI` = 1 [s] must be set.




The block works according to the following formula:

$$Out_{(n)} = Out_{(n-1)} + \frac{SampleTime}{2 \cdot TI} \cdot \left( In_{(n-1)} + In_{(n)} \right)$$

Where:

- `Out` = Integrated value between high and low limits
- `SampleTime` = Sampling time [s]

- `TI` = Integration time constant [s]
- `In` = Input value
- $\texttt{In}_{(n-1)}$ = Last input value
- $\texttt{Out}_{(n-1)}$ = Last output value

You can find additional information on forming the signal status under Forming and outputting the signal status for mathematical blocks (Page 126)

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

There is an example project for the Integral block (APL_Example_xx, xx refers to the language variant) with an application scenario for this block, which explains how the block works.

Application scenario in the example project:

- Process simulation including noise generator (ProcSimC; ProcSimS) (Page 2871)

### Startup characteristics

Use the `Feature` bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

Integral functions (Page 2255)

Integral messaging (Page 2258)

Integral I/Os (Page 2258)

Integral block diagram (Page 2259)

Integral error handling (Page 2257)

Integral modes (Page 2254)

## 13.9.2 Integral modes

### Integral modes

This block does not have any modes.

### See also

Integral block diagram (Page 2259)

Integral I/Os (Page 2258)

Integral messaging (Page 2258)

Integral error handling (Page 2257)

Integral functions (Page 2255)

Description of Integral (Page 2253)

## 13.9.3 Integral functions

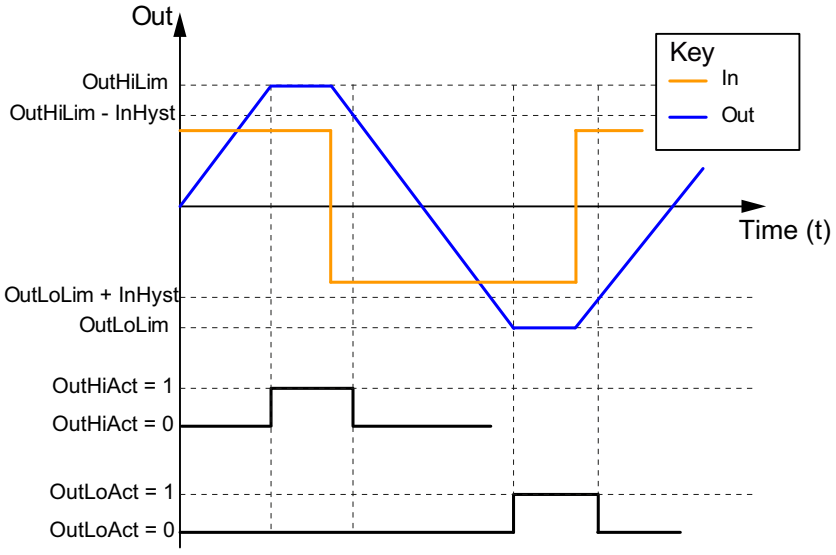
### Functions of Integral

The functions for this block are listed below.

### Monitoring limits

Use the `OutHiLim` and `OutLoLim` I/Os to define the limits of the integrated value `Out`.

If limits are reached or exceeded (`OutHiAct` or `OutLoAct` = 1 I/Os), the output value is set to user-defined limits and output at `Out`.




### Tracking values

You use input parameter `OutTrkOn` = 1 to activate tracking of a value, which is in turn defined at input parameter `OutTrk`.

After tracking mode has been terminated, the block uses the value currently set at `Out` as the value to be integrated.

In tracking mode, the signal status of `Out` is set to the signal status of `OutTrk`.

**Note**

If the integration is stopped, the function (`Hold` = 1) has priority over the tracking.

## Stopping integration

Set input parameter `Hold` = 1 if you want to stop the integration. Limits are no longer monitored and the updating of the `OutHiAct` and `OutLoAct` limit outputs stops. A change to the monitoring limits remains ineffective when integration is stopped. If you start the integration again, the value currently pending at `Out` output parameter is used for the integration process.

## Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 8 | Status output calculation (Page 202) |

## Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for technologic blocks (Page 117).

The signal status for the block is formed using the following parameters and output at the `Out`, `OutHiAct` and `OutLoAct` output parameters:

- `In.ST`
- `Out.ST` Signal status from the last cycle

**Note**

**Special notes for the Integral mathematical block**

Due to its application area (time integral, I-component for the configuration of a controller), the Integral block generates the signal status like the technological blocks.

## See also

Description of Integral (Page 2253)

Integral messaging (Page 2258)

Integral I/Os (Page 2258)

Integral block diagram (Page 2259)

Integral error handling (Page 2257)

Integral modes (Page 2254)

Forming and outputting the signal status for mathematical blocks (Page 126)

## 13.9.4 Integral error handling

### Error handling of Integral

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 11 | The integration time constant `TI` is within the range:<br>`-SampleTime` / 2 < `TI` < `SampleTime` / 2.<br>The integration is stopped. |
| 15 | `OutTrk` > `OutHiLim`<br>`OutTrk` < `OutLoLim` |
| 30 | The value of `In` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out` output. |
| 32 | The value of `OutTrk` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out` output. |
| 33 | The value of `TI` can no longer be displayed in the REAL number field. The integration is stopped. |

### See also

Integral I/Os (Page 2258)

Integral messaging (Page 2258)

Integral functions (Page 2255)

Integral modes (Page 2254)

Description of Integral (Page 2253)

Integral block diagram (Page 2259)

## 13.9.5 Integral messaging

### Messaging

This block does not offer messaging.

### See also

Description of Integral (Page 2253)
Integral functions (Page 2255)
Integral I/Os (Page 2258)
Integral block diagram (Page 2259)
Integral error handling (Page 2257)
Integral modes (Page 2254)

## 13.9.6 Integral I/Os

### I/Os of Integral

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `EN` | 1 = Called block will be processed | BOOL | 1 |
| `Feature` | I/O for additional functions (Page 2255) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| `Hold` | 1 = Integration is held | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `In*` | Analog input value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `InHyst*` | Hysteresis | REAL | 0.0 |
| `OutHiLim` | High limit of output value | REAL | 100.0 |
| `OutLoLim` | Low limit of output value | REAL | 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OutTrkOn | 1 = Output Out tracks the set value (OutTrk I/O) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OutTrk* | Predefined value used for OutTrkOn = 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| TI* | Integral time constant [s] | REAL | 1.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see Integral error handling (Page 2257) | INT | -1 |
| Out | Integral value output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| OutHiAct | 1 = High limit (OutHiLim) reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OutLoAct | 1 = Low limit (OutLoLim) reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### See also

Description of Integral (Page 2253)
Integral messaging (Page 2258)
Integral block diagram (Page 2259)
Integral modes (Page 2254)

## 13.9.7 Integral block diagram

### Integral block diagram

A block diagram is not provided for this block.

**See also**

Integral I/Os (Page 2258)
Integral messaging (Page 2258)
Integral functions (Page 2255)
Integral modes (Page 2254)
Description of Integral (Page 2253)
Integral error handling (Page 2257)

# 13.10 Lag - Low-pass filter

## 13.10.1 Description of Lag

### Object name (type + number) and family

Type + number: FB 1828

Family: Math

### Area of application for Lag

The block is used for the following application:

- Smoothing the Input value (low-pass filter)

### How it works

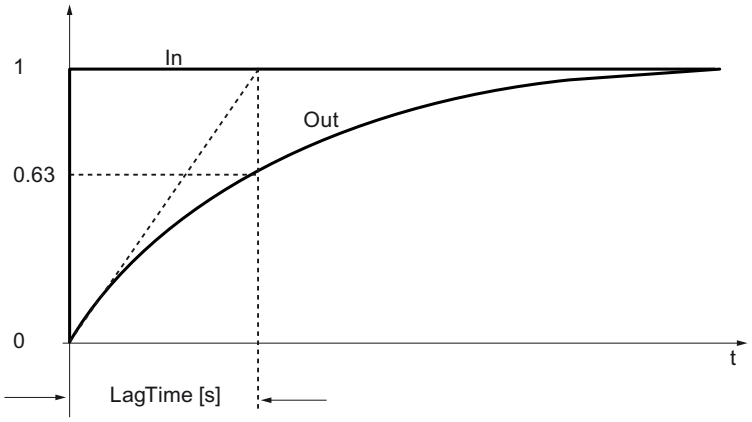
The block smoothes the input variable (`In` input) using a 1. order time delay. This delay time can be configured (`LagTime` connection). The block works according to the following formula:

$$Out_{(n)} = In_{(n)} + (Out_{(n-1)} - In_{(n)}) \cdot e^{\left(\frac{-SampleTime}{LagTime}\right)}$$

Where:

- `Out` = Output value
- `LagTime` = Delay time
- `SampleTime` = Sampling time
- `In` = Input value

The formula only applies to `LagTime` > 0. If `LagTime` = 0, the input is passed directly to the output. If the input value is outside the REAL range limits, the calculation is stopped. If the input value is outside the range limits again, the calculation is resumed automatically.




You can find additional information on forming the signal status under Forming and outputting the signal status for mathematical blocks (Page 126)

## Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB3x). The block is also installed automatically in the startup OB (OB100).

Further addressing is not required.

For the Lag block, the Advanced Process Library contains templates for process tag types as examples and there is a example project (APL_Example_xx, xx designates the language variant) containing different application cases for this block. Several application cases are simulated in the example project and serve to explain how the block works.

Examples of process tag types:

- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 2827)
- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 2825)
- Model-based predictive control (ModPreCon) (Page 2849)
- PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 2823)
- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 2822)
- Ratio control with control loop monitoring through ConPerMon (RatioControl) (Page 2836)
- Ratio control with PIDConR (RatioR) (Page 2839)
- PID controller with Smith predictor (SmithPredictorControl) (Page 2830)
- Cascade control with control loop monitoring through ConPerMon (CascadeControl) (Page 2840)
- Split-range controller with control loop monitoring through ConPerMon (SplitrangeControl) (Page 2833)

Application scenario in the example project:

- Process simulation including noise generator (ProcSimC; ProcSimS) (Page 2871)

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

Lag functions (Page 2263)

Lag messaging (Page 2265)

Lag I/Os (Page 2266)

Lag block diagram (Page 2267)

Lag error handling (Page 2264)

Lag modes (Page 2263)

## 13.10.2 Lag modes

### Lag modes

This block does not have any modes.

### See also

Lag block diagram (Page 2267)

Lag I/Os (Page 2266)

Lag messaging (Page 2265)

Lag error handling (Page 2264)

Lag functions (Page 2263)

Description of Lag (Page 2261)

## 13.10.3 Lag functions

### Functions of Lag

The functions for this block are listed below.

### Hold and restart calculation

You can interrupt the calculation by setting `Hold` = 1 . The output value is frozen. Continue calculation by setting `Hold` = 0 . The calculation is continued with the last output value.

### Reset values

You need to set the `Reset` = 1 I/O if you want to reset the output value back to the input value. The output is reset by a rising 0 - 1 edge.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) section. The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for mathematical blocks (Page 126).

The signal status for the block is formed using the following parameters and output at the `Out` output parameter:

- `In.ST`

### See also

Description of Lag (Page 2261)

Lag messaging (Page 2265)

Lag I/Os (Page 2266)

Lag block diagram (Page 2267)

Lag error handling (Page 2264)

Lag modes (Page 2263)

## 13.10.4 Lag error handling

### Error handling of Lag

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; block is not processed |
| 0 | There is no error. |
| 11 | `LagTime` < 0 |
| 30 | The value of `In` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out` output. |
| 32 | The value of `LagTime` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out` output. |

### See also

Lag block diagram (Page 2267)
Lag I/Os (Page 2266)
Lag messaging (Page 2265)
Lag functions (Page 2263)
Lag modes (Page 2263)
Description of Lag (Page 2261)

## 13.10.5 Lag messaging

### Messaging

This block does not offer messaging.

### See also

Description of Lag (Page 2261)
Lag functions (Page 2263)
Lag I/Os (Page 2266)
Lag block diagram (Page 2267)
Lag modes (Page 2263)
Lag error handling (Page 2264)

## 13.10.6 Lag I/Os

### I/Os of Lag

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `EN` | 1 = Called block will be processed | BOOL | 1 |
| `Feature` | I/O for additional functions (Page 2263) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| `Hold` | 1 = Hold calculation | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `In*` | Analog input value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `LagTime*` | Delay time [s] | REAL | 1.0 |
| `Reset` | 1 = Reset `Out` output to the value of the `In` input | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `SampleTime` | Sampling time [s] (assigned automatically) | REAL | 0.1 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `ENO` | 1 = Block algorithm completed without errors | BOOL | 0 |
| `ErrorNum` | Output of current error number. For error numbers that can be output by this block, see Lag error handling (Page 2264) | INT | -1 |
| `Out` | Delayed value output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### See also

Description of Lag (Page 2261)

Lag messaging (Page 2265)

Lag block diagram (Page 2267)
Lag modes (Page 2263)

## 13.10.7 Lag block diagram

### Lag block diagram

A block diagram is not provided for this block.

### See also

Lag I/Os (Page 2266)
Lag messaging (Page 2265)
Lag error handling (Page 2264)
Lag functions (Page 2263)
Lag modes (Page 2263)
Description of Lag (Page 2261)

# 13.11 MeanTime - Averaging

## 13.11.1 Description of MeanTime

### Object name (type + number) and family

Type + number: FB 1832

Family: Math

### Area of application for MeanTime

The block is used for the following applications:

- Averaging an analog value over a previous, definable period

### How it works

The MeanTime block is used to calculate the time-based mean value of an analog input signal `In` over a previous, definable period (`TimeWindow` input), according to the formula:

`Out` = (`In1` + ... + `Inn`) / (`TimeWindow` / `SampleTime`)

Where:

- `In1` ... `Inn` the n are the detected values used for averaging.
- The time window for the averaging is set at the `TimeWindow` parameter.
- The block determines the number of values to be saved based on the integer part of the `TimeWindow` / `SampleTime` quotient.
- The block can save up to 32 previous values in its internal memory. By interconnecting the `Mem` input parameter with the `Mem` output parameter of the MemR256 block, the memory size can be expanded to 256 previous values. If the time window is longer than memory spaces are available, an internal message is triggered in addition.

If `SampleTime` or `TimeWindow` is changed, the mean time value is reset.

The signal status is passed from the input parameter directly to the output parameter.

You can find additional information on forming the signal status under Forming and outputting the signal status for mathematical blocks (Page 126)

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (all OB3x blocks). The block is also installed automatically in the startup OB (OB 100). When using the MemR256 block, the `Mem` output parameter of the MemR256 block must be interconnected with the `Mem` input parameter of the MeanTime block. The MemR256 block can be inserted in any cyclic interrupt level.

**Note**

The MemR256 block always uses one memory cell less than it is necessary for the time window `TimeWindow`, since the input In itself serves as the memory cell.For example: For a time window that requires 256 memory cells, the memory cell Mem.Val255 on the MemR256 block is not updated.

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

MeanTime functions (Page 2270)
MeanTime messaging (Page 2272)
MeanTime I/Os (Page 2273)
MeanTime block diagram (Page 2274)
MeanTime error handling (Page 2272)
MeanTime modes (Page 2269)

## 13.11.2 MeanTime modes

### MeanTime operating modes

This block does not have any operating modes.

### See also

MeanTime block diagram (Page 2274)
MeanTime I/Os (Page 2273)
MeanTime messaging (Page 2272)

MeanTime error handling (Page 2272)
MeanTime functions (Page 2270)
Description of MeanTime (Page 2268)

## 13.11.3 MeanTime functions

### Functions of MeanTime

The functions for this block are listed below.

### Stopping the calculation of mean values

Mean value calculation can be stopped by setting the `Hold` input. To do this, make the following parameter settings:

- `Hold` = 1 and calculation is stopped. The output value is retained for the duration of this period.
- `Hold` = 0 and calculation is resumed

If the hold functionality is active, the output `HoldAct` will be set by the block.

With `Feature.Bit5` Retain last output value in case of bad input signal status (Page 198), you can specify that the calculation of mean value is stopped and the last output value is retained if the signal status of `In` is bad (`In.ST` = 16#00 or 16#28). In this case, the block sets both the outputs `HoldAct` and `Inbad`.

### Setting a mean value constant

You can set the mean value using the `Reset` input. To do this, make the following parameter setting:

- `Reset` = 0 -> 1

The value at the `In` input is now set directly at the `Out` output. All internal values of the block are also adapted to the input value.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 5 | Retain last output value in case of bad input signal status (Page 198) |

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for mathematical blocks (Page 126).

The signal status for the block is formed using the following parameters and output at the `Out` output parameter:

- `In.ST`

### Increasing the internal memory to 256 previous values

MeanTime can save up to 32 previous values internally. If the ratio of time window and sampling time gets too large, the number of internal memory slots can be expanded to 256 previous values by using the MemR256 block. To do so insert the MemR256 block into a CFC chart. Interconnect the `Mem` output parameter of the MemR256 block with the `Mem` input parameter of the MeanTime block. Each MeanTime block must be interconnected with its own separate MemR256 block. An internal averaging is automatically activated if the time window is still too large. You detect whether internal averaging is active at the `ExcessCnt` output parameter:

- `ExcessCnt` = 0: no internal averaging active.
- `ExcessCnt > 0`: : internal averaging active.
  Without MemR256: ExcessCnt = TimeWindow / SampleTime Div 32 + 1
  Where MemR256: ExcessCnt = TimeWindow / SampleTime Div 256 + 1

When internal averaging is activated, the `In.Value` input is summed using ExcessCnt values (SumIntern) and after ExcessCnt cycles the internal average is stored as a past value at a memory location (1..UsedBuffer). The current value of the `Out` output is then calculated as follows:

- Out:=( ExcessCnt * Sum + SumIntern) / (UsedBuffer * ExcessCnt + Index )
- UsedBuffer: TimeWindow / SampleTime Div ExcessCnt
- Sum: Sum of all storage locations (UsedBuffer)
- SumIntern: Sum of the internal average value from 1 to index
- Index: Running index from 1 to ExcessCnt

**Note**

The ratio of the time window and sampling time is limited internally to 25600. If the time window is set too large, an error number is output at the `ErrorNum` output parameter.

### See also

Description of MeanTime (Page 2268)

MeanTime messaging (Page 2272)

MeanTime I/Os (Page 2273)

MeanTime block diagram (Page 2274)

MeanTime error handling (Page 2272)

MeanTime modes (Page 2269)

## 13.11.4 MeanTime error handling

### Error handling of MeanTime

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed. |
| 0 | There is no error. |
| 30 | The value of `In` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out` output. |
| 31 | The value of `Out` can no longer be displayed in the REAL number field. The `In` value is output unmodified at the `Out` output. |
| 50 | The quotient TimeWindow / SampleTime is greater than 25600. The block calculates with a smaller time window, so that the quotient TimeWindow / SampleTime does not exceed the value 25600. |

### See also

MeanTime block diagram (Page 2274)

MeanTime I/Os (Page 2273)

MeanTime messaging (Page 2272)

Description of MeanTime (Page 2268)

MeanTime modes (Page 2269)

MeanTime functions (Page 2270)

## 13.11.5 MeanTime messaging

### Messaging

This block does not offer messaging.

### See also

Description of MeanTime (Page 2268)

MeanTime functions (Page 2270)

MeanTime I/Os (Page 2273)

MeanTime block diagram (Page 2274)

MeanTime modes (Page 2269)

MeanTime error handling (Page 2272)

## 13.11.6 MeanTime I/Os

I/Os of MeanTime

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `EN` | 1 = Called block will be processed | BOOL | 1 |
| `Feature` | I/O for additional functions (Page 2270) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| `Hold` | 1 = Hold calculation of mean value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `In*` | Analog input value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `Mem` | Input for Mem Block | ANY | - |
| `Reset` | 1 = Reset output `Out` | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `SampleTime` | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| `TimeWindow` | Size of the time window [s] | REAL | 32.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see MeanTime error handling (Page 2272) | INT | -1 |
| ExcessCnt | Counter for internal averaging;<br>0: no internal averaging. All values of In can be saved as previous values with the current TimeWindow.<br>x: internal averaging of In across x cycles. The internal averaging is saved as previous value. | INT | 0 |
| HoldAct | The block is on hold. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| InBad | 1 = Signal status of the analog input In is bad (16#00 or 16#28) | BOOL | 0 |
| Out | Output for the average time | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### See also

Description of MeanTime (Page 2268)

MeanTime messaging (Page 2272)

MeanTime block diagram (Page 2274)

MeanTime modes (Page 2269)

## 13.11.7 MeanTime block diagram

### MeanTime block diagram

This block is not provided a block diagram.

### See also

Description of MeanTime (Page 2268)

MeanTime modes (Page 2269)

MeanTime functions (Page 2270)

MeanTime error handling (Page 2272)

MeanTime messaging (Page 2272)

MeanTime I/Os (Page 2273)

# 13.12 Mul04 - Multiplier with 4 values

## 13.12.1 Description of Mul04

### Object name (type + number) and family

Type + number: FC 360

Family: Math

### Area of application for Mul04

The block is used for the following applications:

- Multiplication of values
- Output of the product for further processing

### How it works

The Mul04 block calculates the product of up to 4 values and returns the result at the `Out` output.

`Out` = `In1` ·...· `Inn` (n ≤ 4),

Where:

`Out` = Product

`In1` ... `In4` = values to be multiplied

The product of all input parameters and the worst input parameter signal status is always output.

The output value is checked to determine if it is within the value range of REAL. If it is outside the value range, the highest or lowest possible REAL value is output.

If the value is not a number `(#NaN)`, the last valid output value is output and the status of the output value is set to 16#28 (if none of the input statuses is worse).

You can find additional information on forming the signal status under Forming and outputting the signal status for mathematical blocks (Page 126)

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Mul04 block, the Advanced Process Library contains templates for process tag types as examples and there is a example project (APL_Example_xx, xx designates the language variant) containing an application case for this block. An application case is simulated in the example project and serves to explain how the block works.

Examples of process tag types:

- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 2827)
- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 2825)

- Model-based predictive control (ModPreCon) (Page 2849)
- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 2822)
- PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 2823)
- Ratio control with PIDConR (RatioR) (Page 2839)
- Ratio control with control loop monitoring through ConPerMon (RatioControl) (Page 2836)
- PID controller with Smith predictor (SmithPredictorControl) (Page 2830)
- Cascade control with control loop monitoring through ConPerMon (CascadeControl) (Page 2840)
- Split-range controller with control loop monitoring through ConPerMon (SplitrangeControl) (Page 2833)

Application scenario in the example project:

- Process simulation including noise generator (ProcSimC; ProcSimS) (Page 2871)

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

### See also

Mul04 functions (Page 2277)

Mul04 messaging (Page 2278)

Mul04 I/Os (Page 2278)

Mul04 block diagram (Page 2279)

Mul04 error handling (Page 2278)

Mul04 modes (Page 2276)

## 13.12.2 Mul04 modes

### Mul04 operating modes

This block does not have any modes.

### See also

Mul04 block diagram (Page 2279)
Mul04 I/Os (Page 2278)
Mul04 messaging (Page 2278)
Mul04 error handling (Page 2278)
Mul04 functions (Page 2277)
Description of Mul04 (Page 2275)

## 13.12.3 Mul04 functions

### Functions of Mul04

The functions for this block are listed below.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for mathematical blocks (Page 126).

The worst signal status for the block is formed by the following parameters and output at the `Out` output parameter:

- `In1.ST`
- `In2.ST`
- `In3.ST`
- `In4.ST`

### See also

Description of Mul04 (Page 2275)
Mul04 messaging (Page 2278)
Mul04 I/Os (Page 2278)
Mul04 error handling (Page 2278)
Mul04 block diagram (Page 2279)
Mul04 modes (Page 2276)

## 13.12.4 Mul04 error handling

### Mul04 error handling

The block does not report any errors.

### See also

Mul04 functions (Page 2277)
Mul04 block diagram (Page 2279)
Mul04 I/Os (Page 2278)
Mul04 messaging (Page 2278)
Description of Mul04 (Page 2275)
Mul04 modes (Page 2276)

## 13.12.5 Mul04 messaging

### Messaging

This block does not offer messaging.

### See also

Description of Mul04 (Page 2275)
Mul04 functions (Page 2277)
Mul04 I/Os (Page 2278)
Mul04 block diagram (Page 2279)
Mul04 modes (Page 2276)
Mul04 error handling (Page 2278)

## 13.12.6 Mul04 I/Os

### Mul04 I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In1 | Value 1 to multiply | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |
| In2 | Value 2 to multiply | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |
| In3 | Value 3 to multiply | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |
| In4 | Value 4 to multiply | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Product version | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### See also

Description of Mul04 (Page 2275)
Mul04 functions (Page 2277)
Mul04 messaging (Page 2278)
Mul04 block diagram (Page 2279)
Mul04 modes (Page 2276)
Mul04 error handling (Page 2278)

## 13.12.7 Mul04 block diagram

### Mul04 block diagram

A block diagram is not provided for this block.

**See also**

Description of Mul04 (Page 2275)
Mul04 modes (Page 2276)
Mul04 functions (Page 2277)
Mul04 error handling (Page 2278)
Mul04 messaging (Page 2278)
Mul04 I/Os (Page 2278)

# 13.13 Mul08 - Multiplier with 8 values

## 13.13.1 Description of Mul08

### Object name (type + number) and family

Type + number: FC 361

Family: Math

### Area of application for Mul08

The block is used for the following applications:

- Multiplication of values
- Output of the product for further processing

### How it works

The Mul08 block calculates the product of up to 8 values and returns the result at the `Out` output.

`Out` = `In1` ·...· `Inn` (n ≤ 8),

Where:

`Out` = Product

`In1` ... `In8` = values to be multiplied

The product of all input parameters and the worst input parameter signal status is always output.

The output value is checked to determine if it is within the value range of REAL. If it is outside the value range, the highest or lowest possible REAL value is output.

If the value is not a number `(#NaN)`, the last valid output value is output and the status of the output value is set to 16#28 (if none of the input statuses is worse).

You can find additional information on forming the signal status under Forming and outputting the signal status for mathematical blocks (Page 126)

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Mul04 (Mul08) block, the Advanced Process Library contains process tag type templates; these serve as examples by providing various application scenarios for this block.

Refer to Description of Mul04 (Page 2275) for more information.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

### See also

Mul08 functions (Page 2282)
Mul08 messaging (Page 2284)
Mul08 I/Os (Page 2284)
Mul08 block diagram (Page 2285)
Mul08 error handling (Page 2283)
Mul08 modes (Page 2282)

## 13.13.2 Mul08 modes

### Mul08 operating modes

This block does not have any modes.

### See also

Mul08 block diagram (Page 2285)
Mul08 I/Os (Page 2284)
Mul08 messaging (Page 2284)
Mul08 error handling (Page 2283)
Description of Mul08 (Page 2281)
Mul08 functions (Page 2282)

## 13.13.3 Mul08 functions

### Functions of Mul08

The functions for this block are listed below.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for mathematical blocks (Page 126).

The worst signal status for the block is formed by the following parameters and output at the `Out` output parameter:

- `In1.ST`
- `In2.ST`
- `In3.ST`
- `In4.ST`
- `In5.ST`
- `In6.ST`
- `In7.ST`
- `In8.ST`

**See also**

Description of Mul08 (Page 2281)
Mul08 messaging (Page 2284)
Mul08 I/Os (Page 2284)
Mul08 block diagram (Page 2285)
Mul08 error handling (Page 2283)
Mul08 modes (Page 2282)

## 13.13.4 Mul08 error handling

### Mul08 error handling

The block does not report any errors.

**See also**

Mul08 block diagram (Page 2285)
Mul08 I/Os (Page 2284)
Mul08 messaging (Page 2284)
Mul08 modes (Page 2282)
Mul08 functions (Page 2282)
Description of Mul08 (Page 2281)

## 13.13.5 Mul08 messaging

### Messaging

This block does not offer messaging.

### See also

Description of Mul08 (Page 2281)
Mul08 functions (Page 2282)
Mul08 I/Os (Page 2284)
Mul08 block diagram (Page 2285)
Mul08 modes (Page 2282)
Mul08 error handling (Page 2283)

## 13.13.6 Mul08 I/Os

### Mul08 I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In1 | Value 1 to multiply | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |
| In2 | Value 2 to multiply | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |
| In3 | Value 3 to multiply | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |
| In4 | Value 4 to multiply | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |
| In5 | Value 5 to multiply | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| In6 | Value 6 to multiply | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |
| In7 | Value 7 to multiply | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |
| In8 | Value 8 to multiply | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 1.0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Product output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### See also

Description of Mul08 (Page 2281)
Mul08 functions (Page 2282)
Mul08 messaging (Page 2284)
Mul08 block diagram (Page 2285)
Mul08 modes (Page 2282)
Mul08 error handling (Page 2283)

## 13.13.7 Mul08 block diagram

### Mul08 block diagram

A block diagram is not provided for this block.

### See also

Mul08 I/Os (Page 2284)
Mul08 messaging (Page 2284)
Mul08 error handling (Page 2283)
Mul08 functions (Page 2282)

Mul08 modes (Page 2282)
Description of Mul08 (Page 2281)

# 13.14 Polygon - Converting the first signal (non-linear)

## 13.14.1 Description of Polygon

### Object name (type + number) and family

Type + number: FB 1881

Family: Math

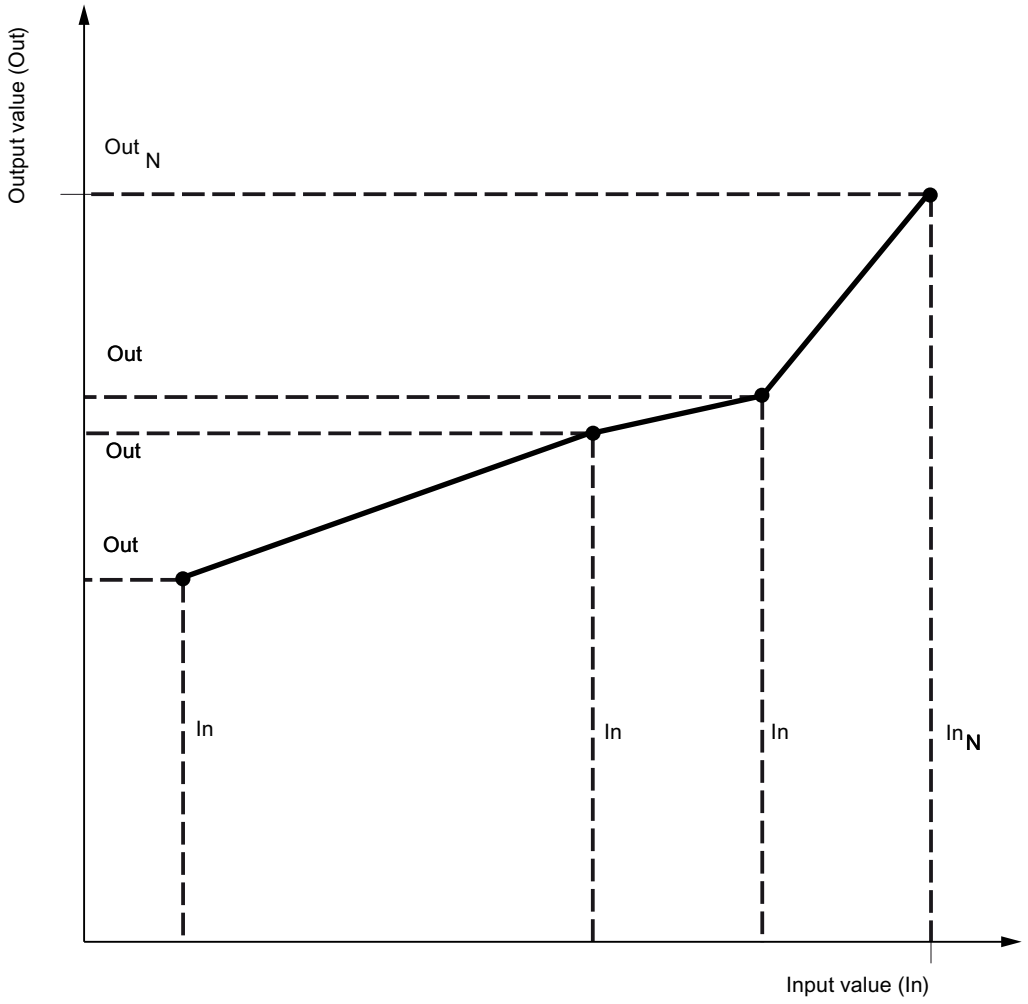
### Area of application for Polygon

The block is used for the following applications:

- Conversion of the input signal according to a non-linear characteristic

### How it works

An input `In` is converted to output `Out` based on a non-linear characteristic with up to 16 interpolation points per polygon block. The number of interpolation points can be increased by cascading multiple blocks. In cascade mode, the `Cascaded = 1` output is set.




| | |
|---|---|
| Num (N) | - Number of intermediate points in the curve<br>Minimum number of points = 2<br>Maximum number of points = 16 |
| In | - Analog input value |
| Out | - Corresponding output value |

After you have defined the N interpolation points (coordinate pairs $In_i$, $Out_i$ with i = 1 ... N in a sequence with no gaps) and configured the number Num, the block operates as follows:

- Interpolation between the interpolation points is linear
- Beyond the end interpolation points, extrapolation is based on the first two or last two interpolation points.

You can find additional information on forming the signal status under Forming and outputting the signal status for mathematical blocks (Page 126)

## Configuration

The block is installed in the CFC editor in a cyclic interrupt OB (OB30 to OB38). The block is installed automatically in OB100.

**Cascade mode**

For the cascading of multiple polygon blocks, the `CasOut` output is interconnected to the `CasIn` input of the following block.

In cascade mode, the interpolation points in all polygon blocks must be configured in ascending order and without gaps. The cascaded polygon blocks must be configured in the same runtime group in ascending run sequence.

The analog input value `In` may only be interconnected at the first polygon block. The `In` input of the next block is tracked to this value.

For the Polygon block, the Advanced Process Library contains templates for process tag types as an example with various application scenarios for this block.

Example of process tag types:

- Source chart for GainSched function block (gain scheduling) (Page 2845)

## Startup characteristics

The block does not have any startup characteristics.

## Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

## See also

Polygon functions (Page 2290)

Polygon messaging (Page 2292)

Polygon I/Os (Page 2293)

Polygon block diagram (Page 2296)

Polygon error handling (Page 2291)

Polygon modes (Page 2290)

## 13.14.2 Polygon modes

### Polygon modes

This block does not have any modes.

### See also

Polygon block diagram (Page 2296)

Polygon I/Os (Page 2293)

Polygon messaging (Page 2292)

Polygon error handling (Page 2291)

Polygon functions (Page 2290)

Description of Polygon (Page 2287)

## 13.14.3 Polygon functions

### Functions of Polygon

The functions for this block are listed below.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for mathematical blocks (Page 126).

The signal status for the block is formed using the following parameters and output at the `Out` output parameter:

- `In.ST`

### Configurable reactions using the `Feature` parameter

An overview of all the reactions that are provided by the `Feature` parameter is available in the section Configurable functions with the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 5 | Limit output Out (Page 154) |
| 6 | Substitute value switch in the event of an error (Page 160) |

### Cascading

- Specifying the number of interpolation points
  When n polygon blocks are cascaded, the interpolation points can be increased by the number set for the Num parameter. The configuration of the interpolation points over all cascaded blocks is subject to the same configuration rules as for an individual block, i.e. the interpolation points must be configured in sequence in ascending order and without gaps.
- Data communication
  The data communication 2 interconnected polygon blocks is made using an output -> input interconnection.

### See also

Description of Polygon (Page 2287)

Polygon messaging (Page 2292)

Polygon I/Os (Page 2293)

Polygon block diagram (Page 2296)

Polygon error handling (Page 2291)

Polygon modes (Page 2290)

## 13.14.4 Polygon error handling

### Error handling of Polygon

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The ErrorNum I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; block is not processed |
| 0 | There is no error. |
| 30 | The value In or SubV_In can no longer be displayed in the REAL- number field. The last valid value is provided at the Out output. |
| 31 | The calculation of the output Out leads to a result that can no longer be displayed in a REAL- numerical field. The last valid value is provided at the Out output. |
| 48 | No valid interconnection to CasIn. The last valid value is provided at the Out output. |

| Error number | Meaning of the error number |
|---|---|
| 49 | `Num` < 2 or `Num` > 16<br>$In_{(N)} \leq In_{(N-1)}$, in the event of an error, the last valid value is output |
| ≥ 1xx | Error in one of the upstream `xx = ErrorNum` block. The last valid value is provided at the `Out` output. |

**Note**

The substitute value `SubV_In` can be output instead of the last valid value via the `Feature.Bit6` = 1 .

Substitute values that cannot be displayed in the REAL- number format have no effect. The last valid value is provided at the `Out` output in this case.

### See also

Polygon block diagram (Page 2296)

Polygon I/Os (Page 2293)

Polygon messaging (Page 2292)

Description of Polygon (Page 2287)

Polygon modes (Page 2290)

Polygon functions (Page 2290)

## 13.14.5 Polygon messaging

### Messaging

This block does not offer messaging.

### See also

Description of Polygon (Page 2287)

Polygon functions (Page 2290)

Polygon I/Os (Page 2293)

Polygon block diagram (Page 2296)

Polygon error handling (Page 2291)

Polygon modes (Page 2290)

## 13.14.6 Polygon I/Os

### I/Os of Polygon

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `CasIn*` | Input for cascade, to be interconnected with the output parameter `CasOut` of the preceding polygon block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| `EN` | 1 = Called block will be processed | BOOL | 1 |
| `Feature` | I/O for additional functions (Page 2290) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| `In*` | Analog input value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `In1` | Interpolation point `In1` | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `In2` | Interpolation point `In2` | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `In3` | Interpolation point `In3` | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `In4` | Interpolation point `In4` | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `In5` | Interpolation point `In5` | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `In6` | Interpolation point `In6` | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `In7` | Interpolation point `In7` | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| `In8` | Interpolation point `In8` | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `In9` | Interpolation point `In9` | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `In10` | Interpolation point `In10` | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `In11` | Interpolation point `In11` | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `In12` | Interpolation point `In12` | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `In13` | Interpolation point `In13` | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `In14` | Interpolation point `In14` | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `In15` | Interpolation point `In15` | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `In16` | Interpolation point `In16` | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `Num` | Number of interpolation points | INT | 16 |
| `Out1` | Sampling point 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `Out2` | Sampling point 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `Out3` | Sampling point 3 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `Out4` | Sampling point 4 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Out5 | Sampling point 5 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Out6 | Sampling point 6 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Out7 | Sampling point 7 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Out8 | Sampling point 8 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Out9 | Sampling point 9 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Out10 | Sampling point 10 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Out11 | Sampling point 11 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Out12 | Sampling point 12 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Out13 | Sampling point 13 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Out14 | Sampling point 14 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Out15 | Sampling point 15 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Out16 | Sampling point 16 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SubV_In* | Substitute value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `Cascaded` | 1 = Block is cascaded | BOOL | 0 |
| `CasOut` | Output parameter for cascade formation, to be interconnected with the input parameter `CasIn` of the following polygon block | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `ENO` | 1 = Block algorithm completed without errors | BOOL | 0 |
| `ErrorNum` | Output of current error number. For error numbers that can be output by this block, see Polygon error handling (Page 2291) | INT | -1 |
| `Out` | Output value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### See also

Description of Polygon (Page 2287)

Polygon messaging (Page 2292)

Polygon block diagram (Page 2296)

Polygon modes (Page 2290)

## 13.14.7 Polygon block diagram

### Polygon block diagram

A block diagram is not provided for this block.

### See also

Polygon I/Os (Page 2293)

Polygon messaging (Page 2292)

Polygon error handling (Page 2291)

Polygon functions (Page 2290)

Polygon modes (Page 2290)

Description of Polygon (Page 2287)

# 13.15 Smooth - Low pass filter

## 13.15.1 Description of Smooth

### Object name (type + number) and family

Type + number: FB 1890

Family: Math

### Area of application for Smooth

The block is used for the following applications:

- Butterworth low-pass filter, 2nd order with maverick detection

### How it works

The block is used as a low pass filter. This filter allows the signal portions with frequencies below the cutoff frequency to pass practically unattenuated, whereas portions with high frequencies are attenuated. The enables you filter out high frequency interference in the signal (for example, signal noise) and smooth the signal.

In comparison to a first order low pass filter, the Butterworth filter has the advantage that the transition from the passing section to the blocking section is sharper in the Bode diagram. If the frequency area of the interference is known, it can be filtered out with minimal influence on the wanted signal.

The maverick (Page 2885) detection monitors adjacent signals. If signal mavericks are detected, they are not processed any further. The block outputs the last valid signal.

The signal status is passed from the input directly to the output.

You can find additional information on forming the signal status under Forming and outputting the signal status for mathematical blocks (Page 126)

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

You then set the filter time constant for the low pass filter to achieve the desired effect.

To set the filter time constant, it is helpful to observe the original and filtered signals in the CFC trend plotter. The filtered signal should be smoothed as required but not too delayed. An increase in the filter time constant increases the smoothing effect but also increases the delay. Typical starting values for the time constant are around ten times the sample rate of the signal.

There is a sample project for the Smooth block (APL_Example_xx, xx refers to the language variant) with an application case for this block:

- Filtering of noisy measured values in a control loop (SigSmoothSim) (Page 2877)

### Startup characteristics

When OB100 is called, the state variables of the Butterworth filter are initialized based on the current process values.

### Status word allocation for `Status1` parameter

The block does not have a status word allocation.

### See also

Smooth functions (Page 2298)
Smooth messaging (Page 2300)
Smooth I/Os (Page 2301)
Smooth block diagram (Page 2302)
Smooth error handling (Page 2299)
Smooth modes (Page 2298)

## 13.15.2 Smooth modes

### Smooth operating modes

This block does not have any modes.

### See also

Smooth block diagram (Page 2302)
Smooth I/Os (Page 2301)
Smooth messaging (Page 2300)
Smooth error handling (Page 2299)
Smooth functions (Page 2298)
Description of Smooth (Page 2297)

## 13.15.3 Smooth functions

### Smooth functions

The block provides the following functions:

- Restart low pass filter
- Activate and deactivate maverick detection

### Restart low pass filter

You can recalculated the coefficients of the Butterworth filter. To do this, you must restart filter (`Restart` = 1).

The filter algorithm is then reinitialized, exactly as it is when the CPU is restarted or a change is made to the count value at the input parameter `TimeConstant`. The coefficients of the Butterworth filter are recalculated and the internal state memory of the filter is initialized so that the `CleanPV` output parameter is equal to the `PV` input parameter.

### Activate and deactivate maverick detection

The maverick detection (`OutlDetOn` = 1) monitors the process value `PV` for the difference between two sequential sample values. These have to be within a tolerance range you have specified (`OutlTreshold`).

If the tolerance is violated, the maverick is set to the most recent valid value. The most recent valid measured value is then held constant by the block for a maximum of `OutlCycles` sample steps. If the maverick continues longer than this, it is accepted as a valid measured value.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for mathematical blocks (Page 126).

The signal status for the block is formed using the following parameters and output at the `CleanPV` output parameter:

- `PV.ST`

### See also

Description of Smooth (Page 2297)

Smooth messaging (Page 2300)

Smooth I/Os (Page 2301)

Smooth block diagram (Page 2302)

Smooth error handling (Page 2299)

Smooth modes (Page 2298)

## 13.15.4 Smooth error handling

### Error handling of Smooth

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 2 | `SampleTime` < 0.001 [s] |
| 30 | The value of `PV` can no longer be displayed in the REAL number field. The last valid value is provided at the `CleanPV` output. |
| 61 | `TimeConstant` < 5 · `SampleTime` |
| 62 | `TimeConstant/SampleTime` > 1000<br>If there is a parametrization error because `ratio = TimeConstant/SampleTime > 1000`, move the function block into another SIMATIC OB with a slower sample time, preferably into the slowest cycle OB30 and recompile the CFC. If this is still not sufficient, you can set the reduction ratio of the CFC runtime group in which the Smooth function block is called. |

### See also

Smooth block diagram (Page 2302)
Smooth I/Os (Page 2301)
Smooth messaging (Page 2300)
Smooth functions (Page 2298)
Smooth modes (Page 2298)
Description of Smooth (Page 2297)

## 13.15.5 Smooth messaging

### Messaging

This block does not offer messaging.

### See also

Description of Smooth (Page 2297)
Smooth functions (Page 2298)
Smooth I/Os (Page 2301)
Smooth block diagram (Page 2302)
Smooth error handling (Page 2299)
Smooth modes (Page 2298)

## 13.15.6 Smooth I/Os

### I/Os of Smooth

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | This parameter currently has no use and is reserved for future functions. | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| FilterOn | 1 = Filter active | BOOL | 1 |
| OutlCycles | Number of sampling steps to bridge mavericks | INT | 3 |
| OutlDetOn | 1 = Maverick detection is activated | BOOL | 0 |
| OutlThreshold | Limit (trigger threshold) for maverick detection | REAL | 10.0 |
| PV* | Analog input (process value) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Restart* | Restart the filter algorithm | BOOL | 1 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 1.0 |
| TimeConstant | Butterworth filter time constant [s] | REAL | 10.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| CleanPV | Output of the corrected process value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum* | Output of current error number. For error numbers that can be output by this block, see Smooth error handling (Page 2299). | INT | 0 |
| OutlDetected | 1 = Maverick detected | BOOL | 0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

**See also**

Description of Smooth (Page 2297)
Smooth functions (Page 2298)
Smooth messaging (Page 2300)
Smooth block diagram (Page 2302)
Smooth modes (Page 2298)

## 13.15.7 Smooth block diagram

**Smooth block diagram**

A block diagram is not provided for this block.

**See also**

Smooth I/Os (Page 2301)
Smooth messaging (Page 2300)
Smooth error handling (Page 2299)
Smooth functions (Page 2298)
Smooth modes (Page 2298)
Description of Smooth (Page 2297)

# 13.16 SqrRoot - Derive the root of a value

## 13.16.1 Description of SqrRoot

### Object name (type + number) and family

Type + number: FC 394

Family: Math

### Area of application for SqrRoot

The block is used to derive the root from a value as follows:

- `Out.Value = SquareRoot(In.Value)`

### How it works

The input value is checked as to whether it is within the REAL value range and is not negative.

- If the input value is not a number (`#NaN`), the last valid output value is displayed at the output. If the input status is not 16#00, the status of the output value is set to 16#28 and `ErrorNum` is set to 30
- If the input value is negative, the last valid output value is displayed at the output. If the input status is not 16#00, the status of the output value is set to 16#28 and `ErrorNum` is set to 30
- If the input value is positive infinity, the the greatest REAL value is displayed at the output. If the input status is not 16#00, the status of the output value is set to 16#28 and `ErrorNum` is set to 30
- 
- The status of the output value is set to 16#28 (if the input status is not below this). ErrorNum is also set to 30.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

**See also**

SqrRoot block diagram (Page 2307)
SqrRoot I/Os (Page 2306)
SqrRoot messaging (Page 2305)
SqrRoot functions (Page 2304)
SqrRoot modes (Page 2304)
Source chart for GainSched function block (gain scheduling) (Page 2845)
PID controller with Smith predictor (SmithPredictorControl) (Page 2830)
Override control (Page 2846)
Forming and outputting the signal status for mathematical blocks (Page 126)

## 13.16.2 SqrRoot modes

### Operating modes of SqrRoot

This block does not have any operating modes.

**See also**

SqrRoot block diagram (Page 2307)
SqrRoot I/Os (Page 2306)
SqrRoot messaging (Page 2305)
SqrRoot functions (Page 2304)
Description of SqrRoot (Page 2303)

## 13.16.3 SqrRoot functions

### Functions of SqrRoot

The functions for this block are listed below.

### Forming the signal status for blocks

The signal status of the block is formed using the following parameters and output at the Out output parameter:

- `Out.ST = In.ST`

**See also**

SqrRoot block diagram (Page 2307)
SqrRoot I/Os (Page 2306)
SqrRoot messaging (Page 2305)
SqrRoot modes (Page 2304)
Description of SqrRoot (Page 2303)
Forming and outputting the signal status for mathematical blocks (Page 126)

## 13.16.4 SqrRoot error handling

### Error handling of SqrRoot

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed. |
| 0 | There is no error. |
| 30 | The input value is a negative number or it cannot be displayed in the REAL number field. |

**See also**

Sub02 block diagram (Page 2312)
Sub02 I/Os (Page 2311)
Sub02 messaging (Page 2310)
Sub02 functions (Page 2309)
Sub02 modes (Page 2309)
Description of Sub02 (Page 2308)

## 13.16.5 SqrRoot messaging

### Messaging

This block does not offer messaging.

**See also**

SqrRoot block diagram (Page 2307)
SqrRoot I/Os (Page 2306)

SqrRoot functions (Page 2304)

SqrRoot modes (Page 2304)

Description of SqrRoot (Page 2303)

## 13.16.6 SqrRoot I/Os

I/Os of SqrRoot

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| IN | Signal status | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of the pending<br>error number. For error numbers that can be output by this block, see SqrRoot error handling (Page 2305). | INT | • -1 |
| Out | Signal status | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### See also

SqrRoot block diagram (Page 2307)

SqrRoot messaging (Page 2305)

SqrRoot functions (Page 2304)

SqrRoot modes (Page 2304)

Description of SqrRoot (Page 2303)

## 13.16.7 SqrRoot block diagram

### SqrRoot block diagram

A block diagram is not provided for this block.

### See also

SqrRoot I/Os (Page 2306)

SqrRoot messaging (Page 2305)

SqrRoot functions (Page 2304)

SqrRoot modes (Page 2304)

Description of SqrRoot (Page 2303)

# 13.17 Sub02 - Subtracting two values

## 13.17.1 Description of Sub02

### Object name (type + number) and family

Type + number: FC 381

Family: Math

### Area of application for Sub02

The block is used for the following applications:

- Subtracting two values

### How it works

The block subtracts one value from another and outputs the subtraction at the `Out` output parameter as follows:

- `Out` = `In1` - `In2`

The worst signal status is also always output at the output parameter.

The output value is checked to determine if it is within the value range of REAL. If it is outside the value range, the highest or lowest possible REAL value is output.

If the value is not a number `(#NaN)`, the last valid output value is output and the status of the output value is set to 16#28 (if none of the input statuses is worse).

You can find additional information on forming the signal status under Forming and outputting the signal status for mathematical blocks (Page 126)

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Sub02 block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Source chart for GainSched function block (gain scheduling) (Page 2845)
- PID controller with Smith predictor (SmithPredictorControl) (Page 2830)
- Override control (Page 2846)

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

### See also

Sub02 block diagram (Page 2312)
Sub02 I/Os (Page 2311)
Sub02 messaging (Page 2310)
Sub02 error handling (Page 2310)
Sub02 functions (Page 2309)
Sub02 modes (Page 2309)

## 13.17.2 Sub02 modes

### Sub02 operating modes

This block does not have any modes.

### See also

Sub02 block diagram (Page 2312)
Sub02 I/Os (Page 2311)
Sub02 messaging (Page 2310)
Sub02 error handling (Page 2310)
Sub02 functions (Page 2309)
Description of Sub02 (Page 2308)

## 13.17.3 Sub02 functions

### Functions of Sub02

The functions for this block are listed below.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for mathematical blocks (Page 126).

The worst signal status for the block is formed by the following parameters and output at the `Out` output parameter:

- `In1.ST`
- `In2.ST`

**See also**

Sub02 block diagram (Page 2312)
Sub02 I/Os (Page 2311)
Sub02 messaging (Page 2310)
Sub02 error handling (Page 2310)
Sub02 modes (Page 2309)
Description of Sub02 (Page 2308)

## 13.17.4 Sub02 error handling

### Sub02 error handling

The block does not report any errors.

**See also**

Sub02 block diagram (Page 2312)
Sub02 I/Os (Page 2311)
Sub02 messaging (Page 2310)
Sub02 functions (Page 2309)
Sub02 modes (Page 2309)
Description of Sub02 (Page 2308)

## 13.17.5 Sub02 messaging

### Messaging

This block does not offer messaging.

**See also**

Sub02 block diagram (Page 2312)
Sub02 I/Os (Page 2311)

Sub02 error handling (Page 2310)
Sub02 functions (Page 2309)
Sub02 modes (Page 2309)
Description of Sub02 (Page 2308)

## 13.17.6 Sub02 I/Os

### Sub02 I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In1 | Value 1 to subtract | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| In2 | Value 2 to subtract | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Product version | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### See also

Sub02 block diagram (Page 2312)
Sub02 messaging (Page 2310)
Sub02 error handling (Page 2310)
Sub02 functions (Page 2309)
Sub02 modes (Page 2309)
Description of Sub02 (Page 2308)

## 13.17.7 Sub02 block diagram

### Sub02 block diagram

A block diagram is not provided for this block.

### See also

Sub02 I/Os (Page 2311)
Sub02 messaging (Page 2310)
Sub02 error handling (Page 2310)
Sub02 functions (Page 2309)
Sub02 modes (Page 2309)
Description of Sub02 (Page 2308)

# 13.18 PowerXY - General power function

## 13.18.1 Description of PowerXY

### Object name (type + number) and family

Type + number: FC 392

Family: Math

### Area of application for PowerXY

The block is used for the following applications:

- Raise one value to the power of another value
- Output of the calculation for further processing

### How it works

This block calculates the output by raising the input variable `In1` to the power of the input variable `In2`.

```
Out = In1^In2
```

The following equation is used for the calculation:

```
If In1 > 0 THEN
Out: = EXP (In2 * LN (In1));
ELSE
Out: = 0;
END_IF;
```

Where:

`In1` = Input value 1

`In1` = Input value 2

`Out` = Output value

`EXP(IN)` = e to the power IN

`LN` = Natural logarithm

The output value is checked to determine if it is within the value range of REAL. If it is outside the value range or If the value is `#NaN`, the last valid output value is output and the status of the output value is set to 16#28 (if none of the input statuses is worse).

If the input parameter is `In1 < =0`, the output value is set to zero until another calculation is permitted mathematically. The worst signal status is also always output at the output parameter.

You can find additional information on forming the signal status under Forming and outputting the signal status for mathematical blocks (Page 126)

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

PowerXY modes (Page 2314)
PowerXY functions (Page 2315)
PowerXY error handling (Page 2315)
PowerXY messaging (Page 2316)
PowerXY I/Os (Page 2316)
PowerXY block diagrams (Page 2317)

## 13.18.2 PowerXY modes

### PowerXY operating modes

This block does not have any modes.

### See also

Description of PowerXY (Page 2313)
PowerXY functions (Page 2315)
PowerXY error handling (Page 2315)
PowerXY messaging (Page 2316)
PowerXY I/Os (Page 2316)
PowerXY block diagrams (Page 2317)

## 13.18.3 PowerXY functions

### Functions of PowerXY

The functions for this block are listed below.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for mathematical blocks (Page 126).

The worst signal status for the block is formed by the following parameter and output at the `Out` output parameter:

- `In1.ST`
- `In2.ST`

### See also

Description of PowerXY (Page 2313)

PowerXY modes (Page 2314)

PowerXY error handling (Page 2315)

PowerXY messaging (Page 2316)

PowerXY I/Os (Page 2316)

PowerXY block diagrams (Page 2317)

## 13.18.4 PowerXY error handling

### Error handling of PowerXY

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks. The following errors can be displayed for this block: Error numbers Overview of error numbers  The ErrorNum I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; block is not processed. |
| 0 | There is no error. |
| 31 | Output can no longer be displayed as REAL number |
| 11 | Input `In1 <= 0` |

### See also

Description of PowerXY (Page 2313)

PowerXY modes (Page 2314)

PowerXY functions (Page 2315)
PowerXY messaging (Page 2316)
PowerXY I/Os (Page 2316)
PowerXY block diagrams (Page 2317)

## 13.18.5 PowerXY messaging

### Messaging

This block does not offer messaging.

### See also

Description of PowerXY (Page 2313)
PowerXY modes (Page 2314)
PowerXY functions (Page 2315)
PowerXY error handling (Page 2315)
PowerXY I/Os (Page 2316)
PowerXY block diagrams (Page 2317)

## 13.18.6 PowerXY I/Os

I/Os of PowerXY

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In1 | Input 1 (base X) | STRUCT | - |
| | | • Value: REAL | 1.0 |
| | | • ST: BYTE | 16#80 |
| In2 | Input 2 (exponent Y) | STRUCT | - |
| | | • Value: REAL | 0.0 |
| | | • ST: BYTE | 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of the pending<br>error number. For error numbers that can be output by this block, see PowerXY error handling (Page 2315). | INT | -1 |
| Out | Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>0.0<br>16#80 |

### See also

Description of PowerXY (Page 2313)

PowerXY modes (Page 2314)

PowerXY functions (Page 2315)

PowerXY messaging (Page 2316)

PowerXY block diagrams (Page 2317)

## 13.18.7 PowerXY block diagrams

### PowerXY block diagram

A block diagram is not provided for this block.

### See also

Description of PowerXY (Page 2313)

PowerXY modes (Page 2314)

PowerXY functions (Page 2315)

PowerXY error handling (Page 2315)

PowerXY messaging (Page 2316)

PowerXY I/Os (Page 2316)

# Analog logic blocks

# 14

## 14.1 CompAn02 - Comparison of two analog values

### 14.1.1 Description of CompAn02

#### Object name (type + number) and family

Type + number: FC 387

Family: LogicAn

#### Area of application for CompAn02

The block is used for the following applications:

- Comparison of the analog input values `In1` and `In2`

#### How it works

The block compares the two analog input values `In1` and `In2` to see if they are "less than", "less than or equal to", "greater than", "greater than or equal to" and "equal".

There is an independent result output parameter for every comparison operation.

These outputs are formed as follows.

- `GT.Value = 1`, if `In1` > `In2`
- `GE.Value = 1`, if `In1` ≥ `In2`
- `EQ.Value = 1`, if `In1` = `In2`
- `LT.Value = 1`, if `In1` < `In2`
- `LE.Value = 1`, if `In1` ≤ `In2`

#### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Further addressing is not required.

#### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

### See also

CompAn02 modes (Page 2320)
CompAn02 functions (Page 2320)
CompAn02 error handling (Page 2321)
CompAn02 messaging (Page 2322)
CompAn02 I/Os (Page 2322)
CompAn02 block diagram (Page 2323)

## 14.1.2 CompAn02 modes

### CompAn02 operating modes

This block does not have any modes.

### See also

Description of CompAn02 (Page 2319)
CompAn02 functions (Page 2320)
CompAn02 messaging (Page 2322)
CompAn02 error handling (Page 2321)
CompAn02 I/Os (Page 2322)
CompAn02 block diagram (Page 2323)

## 14.1.3 CompAn02 functions

### Functions of CompAn02

The functions for this block are listed below.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status of analog logic blocks (Page 120).

### See also

Description of CompAn02 (Page 2319)
CompAn02 modes (Page 2320)
CompAn02 error handling (Page 2321)
CompAn02 messaging (Page 2322)
CompAn02 I/Os (Page 2322)
CompAn02 block diagram (Page 2323)

## 14.1.4 CompAn02 error handling

### CompAn02 error handling

The block does not report any errors.

### Behavior of the block leaving the REAL number field

If one of the two input parameters `In1` or `In2` is outside the REAL number field, it is treated like a REAL number.

### Example

- `In1 = #e+INF` and `In2 < #e+INF: GT/GE := 1`, the other output parameters are then 0
- `In1 = #e-INF` and `In2 > #e-INF: LT/LE := 1`, the other output parameters are then 0
- `In1 = #NAN#` or `In2 = #NAN#:` all output parameters are set to 0

### See also

Description of CompAn02 (Page 2319)
CompAn02 modes (Page 2320)
CompAn02 functions (Page 2320)
CompAn02 messaging (Page 2322)
CompAn02 I/Os (Page 2322)
CompAn02 block diagram (Page 2323)

## 14.1.5 CompAn02 messaging

### Messaging

This block does not offer messaging.

### See also

Description of CompAn02 (Page 2319)
CompAn02 modes (Page 2320)
CompAn02 functions (Page 2320)
CompAn02 error handling (Page 2321)
CompAn02 block diagram (Page 2323)
CompAn02 I/Os (Page 2322)

## 14.1.6 CompAn02 I/Os

### CompAn02 I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `EN` | 1 = Called block will be processed | BOOL | 1 |
| `In1` | Analog input value 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `In2` | Analog input value 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `ENO` | 1 = Block algorithm completed without errors | BOOL | 0 |
| `EQ` | 1= `In1` = `In2` | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| GE | 1 = In1 ≥ In2 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| GT | 1 = In1 > In2 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| LE | 1 = In1 ≤ In2 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| LT | 1 = In1 < In2 | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |

### See also

Description of CompAn02 (Page 2319)
CompAn02 modes (Page 2320)
CompAn02 functions (Page 2320)
CompAn02 error handling (Page 2321)
CompAn02 messaging (Page 2322)
CompAn02 block diagram (Page 2323)

## 14.1.7 CompAn02 block diagram

### CompAn02 block diagram

A block diagram is not provided for this block.

### See also

Description of CompAn02 (Page 2319)
CompAn02 modes (Page 2320)
CompAn02 functions (Page 2320)
CompAn02 error handling (Page 2321)
CompAn02 messaging (Page 2322)
CompAn02 I/Os (Page 2322)

# 14.2 Limit - Limiting an analog value

## 14.2.1 Description of Limit

### Object name (type + number) and family

Type + number: FB 1829

Family: LogicAn

### Area of application for Limit

The block is used for the following applications:

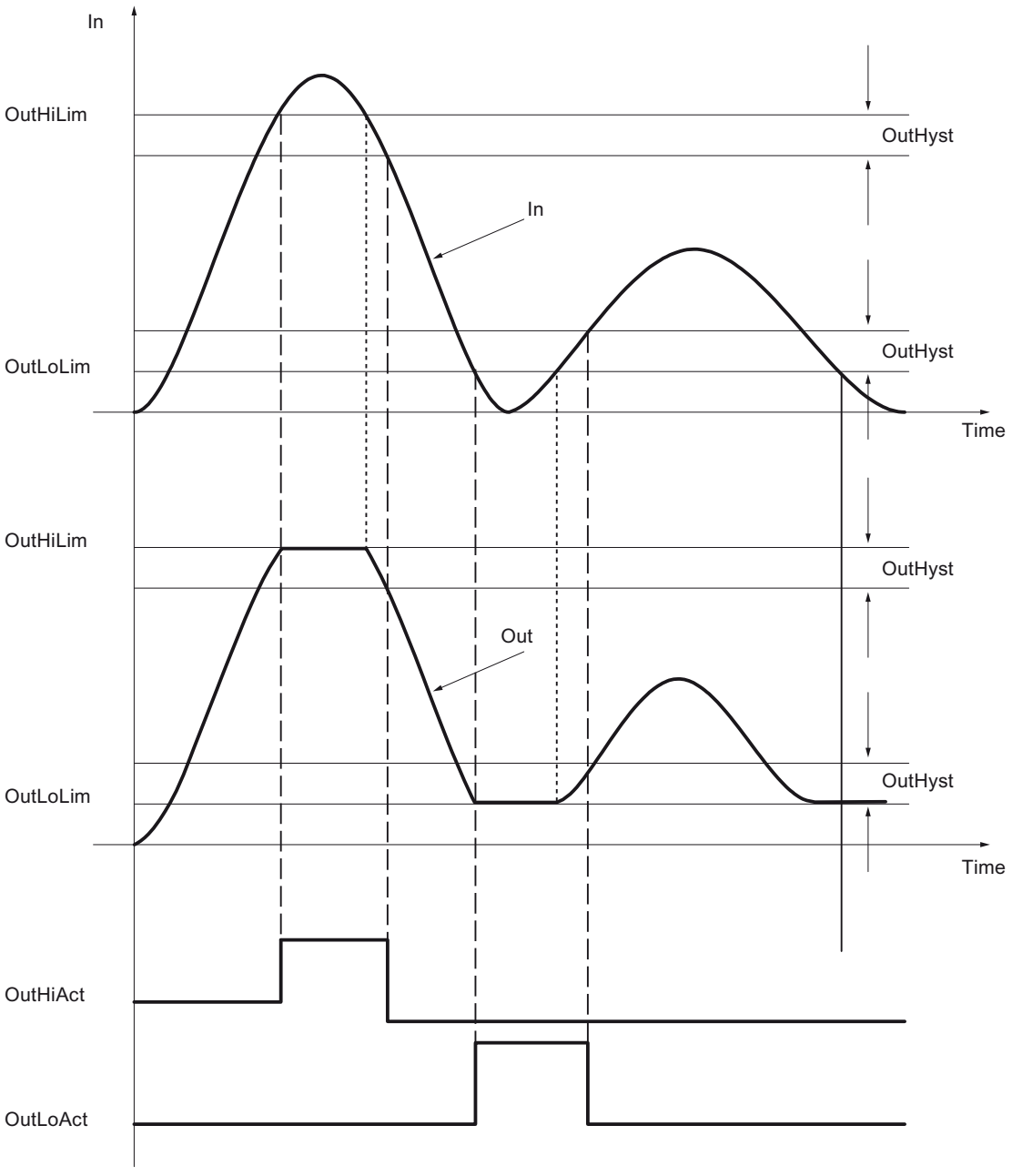
- Limiting of values

### How it works

The Limit block is used to limit an analog value to an adjustable range.

The limits are set at the `OutHiLim` (high limit) and `OutLoLim` (low limit) input parameters. If a limit is violated, the limit you entered is output. The limit violation is also shown at the two output parameters `OutHiAct` (high) or `OutLoAct` (low).

You can configure hysteresis (`OutHyst` input parameter) to suppress signal flutters close to the limits.




| Active conditions for setting limit values | Active conditions for resetting limit values |
|---|---|
| High limit active: In >= OutHiLim | High limit active: In < OutHiLim - OutHyst |
| Low limit active: In <= OutLoLim | Low limit active: In > OutLoLim + OutHyst |

## Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Further addressing is not required.

### Startup characteristics

The block does not have any startup characteristics.

A startup behavior can be reached with a connection of the input parameter `Init` to the output parameter `InitOut` of the Trigger block. For more information, refer to Startup characteristics over Trigger block (Page 75).

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

### See also

Limit block diagram (Page 2329)

Limit I/Os (Page 2328)

Limit messaging (Page 2328)

Limit error handling (Page 2327)

Limit functions (Page 2326)

Limit modes (Page 2326)

## 14.2.2 Limit modes

### Limit modes

This block does not have any modes.

### See also

Limit block diagram (Page 2329)

Limit I/Os (Page 2328)

Limit messaging (Page 2328)

Limit error handling (Page 2327)

Limit functions (Page 2326)

Description of Limit (Page 2324)

## 14.2.3 Limit functions

### Functions of Limit

The function for this block is listed below.

### Initialization

With the input parameter `Init`, the output parameters `OutHiAct` and `OutLoAct` can be initialized:

- `Init` = 0 (default): Normal limitation behavior
  If the input parameter `In` is in the hysteresis range, the output parameters `OutHiAct` and `OutLoAct` will remain unchanged.
- `Init` = 1:
  If the input parameter `In` is in the hysteresis range, the output parameters `OutHiAct` and `OutLoAct` will be reset.

To initialize the output parameters `OutHiAct` and `OutLoAct` after a startup, connect the input parameter `Init` with the output parameter `InitOut` of the Trigger block. For more information, refer to Startup characteristics over Trigger block (Page 75).

### See also

Limit block diagram (Page 2329)

Limit I/Os (Page 2328)

Limit messaging (Page 2328)

Limit error handling (Page 2327)

Limit modes (Page 2326)

Description of Limit (Page 2324)

## 14.2.4 Limit error handling

### Limit error handling

The block does not report any errors.

### See also

Limit block diagram (Page 2329)

Limit I/Os (Page 2328)

Limit messaging (Page 2328)

Limit functions (Page 2326)

Limit modes (Page 2326)

Description of Limit (Page 2324)

## 14.2.5 Limit messaging

### Messaging

This block does not offer messaging.

### See also

Limit block diagram (Page 2329)
Limit I/Os (Page 2328)
Limit error handling (Page 2327)
Limit functions (Page 2326)
Limit modes (Page 2326)
Description of Limit (Page 2324)

## 14.2.6 Limit I/Os

### I/Os of Limit

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 2326) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| In | Analog input value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Init | Initialize function block | BOOL | 0 |
| OutHyst* | Hysteresis in % | REAL | 0.0 |
| OutHiLim | High limit of output value | REAL | 100.0 |
| OutLoLim | Low limit of output value | REAL | 0.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see Limit error handling (Page 2327) | INT | -1 |
| Out | Analog output value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| OutHiAct | 1 = High limit overshoot | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OutLoAct | 1 = Low limit undershoot | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### See also

Limit block diagram (Page 2329)

Limit messaging (Page 2328)

Limit modes (Page 2326)

Description of Limit (Page 2324)

## 14.2.7 Limit block diagram

### Limit  block diagram

A block diagram is not provided for this block.

### See also

Limit I/Os (Page 2328)

Limit messaging (Page 2328)

Limit error handling (Page 2327)

Limit functions (Page 2326)

Limit modes (Page 2326)

Description of Limit (Page 2324)

# 14.3 MuxAn03 - Selection of an analog value to increase availability / safety

## 14.3.1 Description of MuxAn03

### Object name (type + number) and family

Type + number: FB 1860

Family: LogicAn

### Area of application for MuxAn03

The block is used for the following applications:

- Selection of an analog value to increase availability or safety in the input of analog values

### How it works

The block uses up to three process values `PV1` ... `PV3` to determine an output value and outputs this with the corresponding signal status at the `PV` output parameter.

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any special startup characteristics.

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

### See also

MuxAn03 block diagram (Page 2337)

MuxAn03 I/Os (Page 2335)

MuxAn03 messaging (Page 2335)

MuxAn03 error handling (Page 2334)

MuxAn03 functions (Page 2331)

MuxAn03 modes (Page 2331)

## 14.3.2 MuxAn03 modes

### MuxAn03 operating modes

This block does not have any modes.

### See also

MuxAn03 block diagram (Page 2337)
MuxAn03 I/Os (Page 2335)
MuxAn03 messaging (Page 2335)
MuxAn03 error handling (Page 2334)
MuxAn03 functions (Page 2331)
Description of MuxAn03 (Page 2330)

## 14.3.3 MuxAn03 functions

### Functions of MuxAn03

The functions for this block are listed below.

### Selection of output signal

Use the `SelValue` input parameter to select whether the output value is to offer higher availability or higher safety.

### Increasing availability

- **Selection 1 of 2** (SelValue = 0):The block determines the input parameter with the higher priority from the two input parameters PV1 and PV2 based on their signal status. The value determined is written to the PV output parameter.
  If both input parameters have the same signal status, the PV1input parameter is written to the PVoutput parameter.
  **Display of process parameters with bad signal status**
  If the signal status of PV1 is bad (16#00, 16#28), the PV1Bad output is set.
  If the signal status of PV2 is bad (16#00, 16#28), the PV2Bad output is set.
  Regardless of PV3, the PV3Bad output is always set to 0.
  **Mean, median, maximum and minimum**
  They are determined from PV1 and PV2:
  - PV_Mean = (PV1 + PV2) /2
  - PV_Median = PV1
  - PV_Max = Max(PV1, PV2)
  - PV_Min = Min(PV1, PV2)

  The status is formed by PV1.ST and PV2.ST. If the status is the same, it is output at PV_Mean.ST, PV_Median.ST, PV_Max.ST and PV_Min.ST. If the statuses are different, the status with the lower priority is output.
- **Selection 1 of 3** (SelValue = 1): The block determines the input parameter with the highest priority from the three input parameters PV1, PV2 and PV3 based on their signal status. The value determined is written to the PV output parameter.
  If two or more of the input parameters have the same signal status, the one with the lowest index is written to the PV output parameter.
  **Display of process parameters with bad signal status**
  If the signal status of PV1 is bad (16#00, 16#28), the PV1Bad output is set.
  If the signal status of PV2 is bad (16#00, 16#28), the PV2Bad output is set.
  If the signal status of PV3 is bad (16#00, 16#28), the PV3Bad output is set
  **Mean, median, maximum and minimum**
  They are determined from PV1, PV2 and PV3:
  - PV_Mean = (PV1 + PV2 + PV3) /3
  - PV_Median = mean PV value of PV1, PV2 and PV3. If 2 of the 3 PV values are the same, the value of the same PVs is output as the mean PV value
    Examples:
    PV1 =1, PV2 =2, PV3 =3: thus PV_Median =2.
    PV1 =1, PV2 =1, PV3 =3: thus PV_Median =1.
  - PV_Max = Max(PV1, PV2, PV3)
  - PV_Min = Min(PV1, PV2, PV3)
    The status is formed by PV1.ST, PV2.ST and PV3.ST. If the status is the same, it is output at PV_Mean.ST, PV_Max.ST and PV_Min.ST. If the statuses are different, the status with the second highest priority is output.

## Increasing safety

- **Selection 2 of 2** (SelValue = 2): The block determines whether the two input parameters PV1 and PV2 have the same signal status and whether they deviate from one another by no more than the setting at input parameter PlDiff. Only then does the PV output parameter also have this signal status. The analog value of PVis set to a value of PV1. If PV1 and PV2 deviate from one another by more than PlDiff, the PV signal status is set to "Bad, device-related". The analog value of PVis set to a value of PV1.
  If PV1 and PV2 do not deviate from one another by more than PlDiff, but have a different signal state, the PV output parameter is generated from the lower priority signal status of the two input parameters and the associated analog value.
  **Display of non-plausible or bad process parameters**
  If the process value PV1 is non-plausible or bad (16#00, 16#28), the PV1Bad output is set.
  If the process value PV2 is non-plausible or bad (16#00, 16#28), the PV2Bad output is set.
  Regardless of PV3, the PV3Bad output is always set to 0
  **Mean, median, maximum and minimum**
  They are determined from PV1 and PV2. If PV1 and PV2 deviate from one another by more than PlDiff, the signal status of PV_Mean, PV_Median, PV_Max and PV_Min is set to "Bad, device-related" (16#00) and the process values of PV_Mean, PV_Median, PV_Max and PV_Min are frozen. If the absolute deviation is less than or equal to PlDiff, the mean value, maximum and minimum are calculated as follows:
  - PV_Mean = (PV1 + PV2) /2
  - PV_Median = PV1
  - PV_Max = Max(PV1, PV2)
  - PV_Min = Min(PV1, PV2)

  The status is formed by PV1.ST and PV2.ST. If the status is the same, it is output at PV_Mean.ST, PV_Max.ST and PV_Min.ST. If the statuses are different, the status with the lower priority is output.
- **Selection 2 of 3** (SelValue = 3): The block determines whether two of the three input parameters PV1, PV2 and PV3 have the same high priority signal status and whether they deviate from one another by no more than the setting at input parameter PlDiff. Only if this is the case, does the PV output parameter also have this signal status. The analog value of PV is set to the value of the input parameter with the lowest index (PV1 or PV2) of the values determined.
  If all input parameters PV1, PV2 and PV3 deviate from one another by more than PlDiff, the signal status of PV is set to "Bad, device-related". The analog value of PVis set to the value of the PV1 input parameter.
  If at least two input parameters do not deviate from one another by more than PlDiff, but have a different signal state, the value from these input parameters with the signal status of the second highest priority is written to the PV output parameter.
  **Display of non-plausible or bad process parameters**
  If the process value PV1 is non-plausible or bad (16#00, 16#28), the PV1Bad output is set.
  If the process value PV2 is non-plausible or bad (16#00, 16#28), the PV2Bad output is set.
  If the process value PV3 is non-plausible or bad (16#00, 16#28), the PV3Bad output is set.
  **Mean value, maximum and minimum**
  They are determined from PV1, PV2 and PV3. An internal mean value is calculated first from PV1, PV2 and PV3.
  This mean value is used to determine the deviations for each PV value. If all absolute deviations are less than or equal to PlDiff, the mean, median, maximum and minimum are

formed from PV1, PV2 and PV3 and the status with the second highest priority is output (see section Increasing availability, selection 1 to 3).
If at least one of the deviations is outside of PlDiff, the PV with the greatest deviation is excluded from further calculation.
If the two remaining process values (PVx1, PVx2) deviate from one another by more than PlDiff, the signal status of PV_Mean, PV_Median, PV_Max and PV_Min is set to "Bad, device-related" (16#00) and the process values of PV_Mean, PV_Median, PV_Max and PV_Min are frozen. If the absolute deviation is less than or equal to PlDiff, the mean value, maximum and minimum are calculated as follows:

- PV_Mean = (PVx1 + PVx2) /2
- PV_Median = PVx1
- PV_Max = Max(PVx1, PVx2)
- PV_Min = Min(PVx1, PVx2)

The status is formed by PVx1.ST and PVx2.ST. If the status is the same, it is output at PV_Mean.ST, PV_Median, PV_Max.ST and PV_Min.ST. If the statuses are different, the status with the lower priority is output.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for blocks with configurable status prioritization (Page 122).

### See also

MuxAn03 block diagram (Page 2337)
MuxAn03 I/Os (Page 2335)
MuxAn03 messaging (Page 2335)
MuxAn03 error handling (Page 2334)
MuxAn03 modes (Page 2331)
Description of MuxAn03 (Page 2330)

## 14.3.4 MuxAn03 error handling

### MuxAn03 error handling

Parameter assignment errors are handled as follows:

- If the input parameter is SelValue < 0, the parameter is automatically set to SelValue = 0.
- If the input parameter is SelValue > 3, the parameter is automatically set to SelValue = 3 .
- If the input parameter is SelPrio < 0, the parameter is automatically set to SelPrio = 0 .
- If the input parameter is SelPrio > 7, the parameter is automatically set to SelPrio = 7 .

An error number is not output in any of these cases.

### See also

MuxAn03 block diagram (Page 2337)
MuxAn03 I/Os (Page 2335)
MuxAn03 messaging (Page 2335)
MuxAn03 functions (Page 2331)
MuxAn03 modes (Page 2331)
Description of MuxAn03 (Page 2330)

## 14.3.5 MuxAn03 messaging

### Messaging

This block does not offer messaging.

### See also

MuxAn03 block diagram (Page 2337)
MuxAn03 I/Os (Page 2335)
MuxAn03 error handling (Page 2334)
MuxAn03 functions (Page 2331)
MuxAn03 modes (Page 2331)
Description of MuxAn03 (Page 2330)

## 14.3.6 MuxAn03 I/Os

MuxAn03 I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| PV1 | Process value 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV2 | Process value 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV3 | Process value 3 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PlDiff | Plausibility check value | REAL | 0.0 |
| SelPrio* | Setting of the prioritization for forming the best signal status | INT | 6 |
| SelValue* | Selection criterion for the search | INT | 0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| PV | Process value determined | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_Max | Process value maximum | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_Mean | Process value mean | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_Median | Process value median | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_Min | Process value minimum | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV1Bad | Process value PV1 is not plausible or bad (16#00 or 16#28) | BOOL | 0 |
| PV2Bad | Process value PV2 is not plausible or bad (16#00 or 16#28) | BOOL | 0 |
| PV3Bad | Process value PV3 is not plausible or bad (16#00 or 16#28) | BOOL | 0 |

## See also

MuxAn03 block diagram (Page 2337)
MuxAn03 messaging (Page 2335)
MuxAn03 error handling (Page 2334)
MuxAn03 functions (Page 2331)

MuxAn03 modes (Page 2331)

Description of MuxAn03 (Page 2330)

## 14.3.7 MuxAn03 block diagram

### MuxAn03 block diagram

A block diagram is not provided for this block.

### See also

MuxAn03 I/Os (Page 2335)

MuxAn03 messaging (Page 2335)

MuxAn03 error handling (Page 2334)

MuxAn03 functions (Page 2331)

MuxAn03 modes (Page 2331)

Description of MuxAn03 (Page 2330)

# 14.4 MuxAn08 - Selection of an analog value to increase availability / safety

## 14.4.1 Description of MuxAn08

### Object name (type + number) and family

Type + number: FB 1831

Family: LogicAn

### Area of application of MuxAn08

The block is used for the following applications

- Selection of an analog value to increase availability or safety in the input of analog values

### How it works

The block uses up to eight `PV1 ... PV8` process values to determine an output value and outputs this with the corresponding signal status at the PV output parameter.

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any special startup characteristics.

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

## 14.4.2 MuxAn08 modes

### MuxAn08 operating modes

This block does not have any operating modes.

## 14.4.3 MuxAn08 functions

### Functions of MuxAn08

The functions for this block are listed below.

### Selection of output signal

Use the `SelValue` input parameter to select whether the output value is to offer higher availability or higher safety.

### Increasing availability

- **Selection 1of 2, 1 of 3, ... 1 of 8** (`SelValue` = 0..6):
  The input parameter `SelValue` defines the relevant input parameters of the process values to be considered for the determination of PV.

| SelValue | Relevant input parameter | Irrelevant input parameter |
|---|---|---|
| 0 | PV1..PV2 | PV3..PV8 |
| 1 | PV1..PV3 | PV4..PV8 |
| 2 | PV1..PV4 | PV5..PV8 |
| 3 | PV1..PV5 | PV6..PV8 |
| 4 | PV1..PV6 | PV7..PV8 |
| 5 | PV1..PV7 | PV8 |
| 6 | PV1..PV8 | - |

The block determines the highest priority of the two input parameters from the relevant input parameters based on their signal status on and writes it to the output parameter PV.
If several input parameters have the same level of signal status priority, the one with the lowest index is written to the output parameter.
**Display of process parameters with "bad" signal status**
When the relevant input parameters (`PV1`, `PV2`, .., `PV8`) have a "bad" signal status (`16#00` or `16#28`), the corresponding output parameters (`PV1Bad`, `PV2Bad`..`PV8Bad`) are set. The output parameters of the irrelevant input parameters are always reset.
**Mean value, maximum and minimum**

- PV_Mean = (PV1 + PV2 + ... +PVn) / n
- PV_Max = Max(PV1, PV2... PVn)
- PV_Min = Min(PV1, PV2... PVn)
  n depends on SelValue
  The status is formed from the `PV1..PV`n input parameters. If the status is the same, it is output at the `PV_Mean`, `PV_Max` and `PV_Min` output parameters. If the states are different, the status with the second highest priority is output at the `PV_Mean`, `PV_Max` and `PV_Min` output parameters.

## Increasing safety

- **Selection 2 of 2..8** (`SelValue` = 7..13):
  The input parameter `SelValue` defines the relevant input parameters of the process values to be considered for the determination of PV.

| SelValue | Relevant input parameter | Irrelevant input parameter |
|---|---|---|
| 7 | PV1..PV2 | PV3..PV8 |
| 8 | PV1..PV3 | PV4..PV8 |
| 9 | PV1..PV4 | PV5..PV8 |
| 10 | PV1..PV5 | PV6..PV8 |
| 11 | PV1..PV6 | PV7..PV8 |
| 12 | PV1..PV7 | PV8 |
| 13 | PV1..PV8 | - |

The block determines whether two of the relevant input parameters have the same level of signal status priority and whether they deviate from one another by no more than the setting at input parameter `PlDiff`. Only then does the `PV` output parameter also have this signal status. The analog value of `PV` is set to the value of the input parameter with the lowest index of the values determined.
If all input parameters deviate from one another by more than `PlDiff`, the signal status of `PV` is set to "Bad, device-related". The analog value of `PV`is set to the value of the `PV1` input parameter.
If the least two input parameters do not deviate from one another by more than `PlDiff`, but have a different signal state, the value with the signal status of the second highest priority from these input parameters is written to the `PV` output parameter.
**Display of process parameters with "bad" signal status**
When the relevant input parameters (`PV1`, `PV2`, .., `PV8`) are not plausible or have a "bad" signal status (`16#00` or `16#28`), the corresponding output parameters (`PV1Bad`, `PV2Bad`..`PV8Bad`) are set. The corresponding output parameters of the irrelevant input parameters are always reset.
**Mean value, maximum and minimum**
The `PV` values are plausible if the absolute values of the deviation are less than or equal to the `PlDiff`.
If the absolute deviation is less than or equal to `PlDiff`, the mean value, maximum and minimum are calculated as follows:

1. Determination of the mean from `PV1.. PV`n
   `n` depends on `SelValue`
2. Determination of the deviations `ABS(PVi` − Mean value)
3. If all `PV` are plausible, then the mean, maximum and minimum are determined, as described in section "Increasing availability `SelValue = 0..6`".

4. If one or more PV are not plausible, repeat the calculation of the mean starting with 1. without PV input with the highest deviation. If there is more than one PV input with equally high deviation, only the first PV input is not used for further calculation.
5. If only 2 PV inputs remain and they are plausible, then the mean, maximum and minimum are determined, as described in section "Increasing availability SelValue = 0". When they are not plausible, the mean value, maximum and minimum are determined as follows:

```
PV_Max.Value: Keep old value
PV_Min.Value: Keep old value
PV_Mean.Value: Keep old value
PV_Max.ST = 16#00;
PV_Min.ST = 16#00;
PV_Mean.ST = 16#00;
```

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for blocks with configurable status prioritization (Page 122).

## 14.4.4 MuxAn08 error handling

### Error handling of MuxAn08

Parameter assignment errors are handled as follows:

- If the input parameter is SelValue < 0, the parameter is automatically set to SelValue = 0.
- If the input parameter is SelValue > 13, the parameter is automatically set to SelValue = 13 .
- If the input parameter is SelPrio < 0, the parameter is automatically set to SelPrio = 0 .
- If the input parameter is SelPrio > 7, the parameter is automatically set to SelPrio = 7 .

An error number is not output in any of these cases.

## 14.4.5 MuxAn08 messaging

### Messaging

This block does not offer messaging.

## 14.4.6 MuxAn08 I/Os

I/Os of MuxAn08

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| PV1 | Process value 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV2 | Process value 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV3 | Process value 3 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV4 | Process value 4 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV5 | Process value 5 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV6 | Process value 6 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV7 | Process value 7 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV8 | Process value 8 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PlDiff | Plausibility check value | REAL | 0.0 |
| SelPrio* | Setting of the prioritization for forming the best signal status | INT | 6 |
| SelValue* | Selection criterion for the search | INT | 0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| PV | Process value determined | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_Mean | Process value mean | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_Max | Process value maximum | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_Min | Process value minimum | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV1Bad | Process value PV1 is not plausible or bad (16#00 or 16#28) | BOOL | 0 |
| PV2Bad | Process value PV2 is not plausible or bad (16#00 or 16#28) | BOOL | 0 |
| PV3Bad | Process value PV3 is not plausible or bad (16#00 or 16#28) | BOOL | 0 |
| PV4Bad | Process value PV4 is not plausible or bad (16#00 or 16#28) | BOOL | 0 |
| PV5Bad | Process value PV5 is not plausible or bad (16#00 or 16#28) | BOOL | 0 |
| PV6Bad | Process value PV6 is not plausible or bad (16#00 or 16#28) | BOOL | 0 |
| PV7Bad | Process value PV is not plausible or bad (16#00 or 16#28) | BOOL | 0 |
| PV8Bad | Process value PV8 is not plausible or bad (16#00 or 16#28) | BOOL | 0 |

## 14.4.7 MuxAn08 block diagram

### MuxAn08 block diagram

A block diagram is not provided for this block.

# 14.5 RateLim - Signal ramp

## 14.5.1 Description of RateLim

### Object name (type + number) and family

Type + number: FB 1882

Family: LogicAn

### Area of application for RateLim

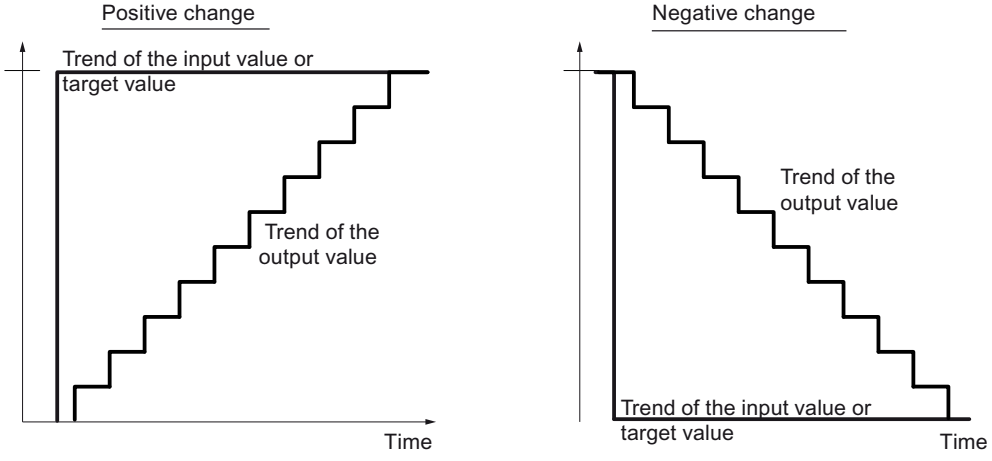
The block is used for the following applications:

- Limitation of the slope of an analog signal
- Ramped approach to a target value

### How it works

Slope limit: Starting from the current output value, the new output value is calculated in such a way that it does not exceed a specified positive or negative slope.

Ramp function: The current output value can be ramped up to a target value. The ramp slope can be configured by a time period or by a positive or negative ramp slope.




The input value is forwarded directly when the gradient and ramp function are switched off (`RmpOn.Value = 0`, `RateOn.Value = 0`): `Out.Value = In.Value`.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB3x). The block is then installed automatically in startup OB (OB100).

Further addressing is not required.

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

RateLim functions (Page 2345)

RateLim messaging (Page 2348)

RateLim I/Os (Page 2349)

RateLim block diagram (Page 2351)

RateLim error handling (Page 2348)

RateLim modes (Page 2345)

## 14.5.2 RateLim modes

### RateLim operating modes

This block does not have any modes.

### See also

RateLim block diagram (Page 2351)

RateLim I/Os (Page 2349)

RateLim messaging (Page 2348)

RateLim error handling (Page 2348)

RateLim functions (Page 2345)

Description of RateLim (Page 2344)

## 14.5.3 RateLim functions

### Functions of RateLim

The functions for this block are listed below.

### Limitation of the slope of the analog signal `In.Value`

The block calculates the gradient of the input signal over time and compares it with the `UpRaLim` (positive change) `DnRaLim` (negative change) limits. See the table below.

- If the slope exceeds the amount of the respective limit (`UpRaLim` or `DnRaLim`), the output `Out` is corrected only by the valid limit and the corresponding limit display `UpRaAct` = 1 or `DnRaAct` = 1 is set.
- If the gradient is within the valid range, the input value is transferred (`In` = `Out`) and `UpRaAct` = 1 or `DnRaAct` = 1 are reset.
- If the corresponding limit is 0 (`UpRaLim` with positive gradient or `DnRaLim` with negative gradient), then the input value `In` is written directly to the output `Out`.

| RateOn | ΔIn / Δt | Meaning | Output `Out` | UpRaAct | DnRaAct |
|---|---|---|---|---|---|
| 1 | < \|`DnRaLim`\| | Input value dropping too fast | `Out` - (\|`DnRaLim`\| · `SampleTime`) * Time factor | 0 | 1 |
| 1 | \|`DnRaLim`\| to `UpRaLim` | Change speed `In` is permitted | `In` | 0 | 0 |
| 1 | > \|`UpRaLim`\| | Input value `In` rising too fast | `Out` + (\|`UpRaLim`\| · `SampleTime`) * Time factor | 1 | 0 |
| 0 | - | `RateOn` is deactivated | `In` | 0 | 0 |

In this case, the time factor is formed from the `TimeFactor` parameter.

| `TimeFactor` | Time factor |
|---|---|
| 0 | 1 |
| 1 | 1/60 |
| 2 | 1/(60*60) |

### Ramp function

Starting from the current output value `Out.Value`, the output value can be ramped to the target value `RmpTarget` when the ramp function is switched on or the target value is changed.

You can use the `RmpModTime` input parameter to specify whether the ramp is defined by time or gradients.

- If you select time (`RmpModTime = 1`): The gradient of the ramp is calculated automatically by the block so that after the start (`RmpOn.Value = 0 -> 1`) or a change of the target value (`RmpTarget.Value`), the target value is reached after the configured time (`RmpTime`). The unit of the ramp time (`RmpTime`) depends on `TimeFactor`.
- If you select gradients (`RmpModTime = 0`): The ramp slope corresponds to the configured rates of change `UpRaLim` (positive) or `DnRaLim` (negative).

When the output value has reached the target, the output value remains at the target value as long as the ramp function remains enabled.

### Switching the ramp function and slope limiting function on or off

The ramp function is switched off with the `RmpOn = 0` input; the slope limiting function is switched off with the `RateOn` = 0 input. If both are switched off, the `In` input value is written directly to the `Out` output. Limits are no longer monitored. If both functions are switched on, the ramp function has priority.

### Stopping the caculation of the ramp output value

Set `Hold` = 1 to interrupt the calculation of the ramp output value. The output value `Out` is held and the output parameters `UpRaAct` and `DnRaAct` are set to 0. The status of `Out`, `UpRaAct`, and `DnRaAct` are realized.

Set `Hold` = 0 to continue the calculation of the ramp output value. The calculation is continued with the last output value.

### Time base of the gradient limits

The gradient limits are set at the `UpRaLim` and `DnRaLim` parameters depending on the `TimeFactor`.

`TimeFactor = 0`: Unit of the gradient limiting is Unit/Second

`TimeFactor = 1`: Unit of the gradient limiting is Unit/Minute

`TimeFactor = 2`: Unit of the gradient limiting is Unit/Hour

### Forming the signal status for blocks

The signal status of `Out` is formed from:

```
RmpOn =1: Out.ST := RmpTarget.ST
```

Otherwise: `Out.ST := In.ST`

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) .

The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 8 | Unit for the rate of change (Page 158) |

### See also

Description of RateLim (Page 2344)

RateLim messaging (Page 2348)

RateLim I/Os (Page 2349)

RateLim block diagram (Page 2351)

RateLim modes (Page 2345)

RateLim error handling (Page 2348)

## 14.5.4 RateLim error handling

### Error handling of RateLim

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 30 | The value of `In` can no longer be displayed in the REAL number field. |
| 43 | Incorrect time unit set at `TimeFactor`. |

### See also

RateLim block diagram (Page 2351)

RateLim I/Os (Page 2349)

RateLim messaging (Page 2348)

RateLim modes (Page 2345)

Description of RateLim (Page 2344)

RateLim functions (Page 2345)

## 14.5.5 RateLim messaging

### Messaging

This block does not offer messaging.

### See also

Description of RateLim (Page 2344)

RateLim functions (Page 2345)

RateLim I/Os (Page 2349)

RateLim block diagram (Page 2351)

RateLim error handling (Page 2348)

RateLim modes (Page 2345)

## 14.5.6 RateLim I/Os

I/Os of RateLim

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `DnRaLim` | Maximum negative change in the output value in units/s or %/s | REAL | 3.0 |
| `EN` | 1 = Called block will be processed | BOOL | 1 |
| `Feature` | I/O for additional functions (Page 2345) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| `Hold` | Hold calculation | STRUCT<br>• Value: BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| `In*` | Analog input value | STRUCT<br>• Value: REAL<br>• ST:BYTE | -<br>• 0.0<br>• 16#80 |
| `InScale` | Scaling of the measuring range as a structure | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| `RateOn` | 1 = Switch on gradient function | STRUCT<br>• Value: BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| `RmpModTime` | 1 = Use time (`RmpTime`) for the ramp function<br>0 = Use gradients (`UpRaLim`, `DnRaLim`) for the ramp function | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RmpOn | 1 = Switch on ramp function for target value RmpTarget | STRUCT<br>• Value: BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| RmpTarget | Target value for the ramp function | STRUCT<br>• Value: REAL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| RmpTime* | Duration [unit depends on TimeFactor] for ramp function from the current Out to RmpTarget | REAL | 0.0 |
| SampleTime | Sampling time [s] (assigned automatically) | REAL | 0.1 |
| TimeFactor | Time unit:<br>0 = Seconds<br>1 = Minutes<br>2 = Hours | INT | 0 |
| UpRaLim | Maximum positive change in the output value in units/s or %/s | REAL | 3.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| DnRaAct | 1 = Negative change in the output value | STRUCT<br>• Value: BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see RateLim error handling (Page 2348) | INT | -1 |
| Out | Output value | STRUCT<br>• Value: REAL<br>• ST:BYTE | -<br>• 0.0<br>• 16#80 |
| UpRaAct | 1 = Positive change in the output value | STRUCT<br>• Value: BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |

### See also

Description of RateLim (Page 2344)
RateLim messaging (Page 2348)
RateLim block diagram (Page 2351)
RateLim modes (Page 2345)

## 14.5.7 RateLim block diagram

### RateLim block diagram

A block diagram is not provided for this block.

### See also

RateLim I/Os (Page 2349)
RateLim messaging (Page 2348)
RateLim error handling (Page 2348)
RateLim functions (Page 2345)
RateLim modes (Page 2345)
Description of RateLim (Page 2344)

# 14.6 RedAn02 - 1 out of 2 selection for redundant analog values

## 14.6.1 Description of RedAn02

### Object name (type + number) and family

Type + number: FC 385

Family: LogicAn

### Area of application for RedAn02

The block is used for the following applications:

- 1 out of 2 selection for redundant analog values

### How it works

The block selects from two input values the one with the best signal status and outputs it at the output `Out`. In addition, the outputs `SimAct`, `Uncertain` and `LossRed` are set according to the signal status.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

RedAn02 modes (Page 2353)

RedAn02 functions (Page 2353)

RedAn02 error handling (Page 2354)

RedAn02 messaging (Page 2354)

RedAn02 I/Os (Page 2354)

RedAn02 block diagram (Page 2356)

## 14.6.2 RedAn02 modes

### RedAn02 operating modes

This block does not have any modes.

### See also

Description of RedAn02 (Page 2352)
RedAn02 functions (Page 2353)
RedAn02 error handling (Page 2354)
RedAn02 messaging (Page 2354)
RedAn02 I/Os (Page 2354)
RedAn02 block diagram (Page 2356)

## 14.6.3 RedAn02 functions

### Functions of RedAn02

The functions for this block are listed below.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status of redundancy blocks (Page 121).

### See also

Description of RedAn02 (Page 2352)
RedAn02 modes (Page 2353)
RedAn02 error handling (Page 2354)
RedAn02 messaging (Page 2354)
RedAn02 I/Os (Page 2354)
RedAn02 block diagram (Page 2356)

## 14.6.4 RedAn02 error handling

### RedAn02 error handling

The block does not report any errors.

### See also

Description of RedAn02 (Page 2352)
RedAn02 modes (Page 2353)
RedAn02 functions (Page 2353)
RedAn02 messaging (Page 2354)
RedAn02 I/Os (Page 2354)
RedAn02 block diagram (Page 2356)

## 14.6.5 RedAn02 messaging

### Messaging

This block does not offer messaging.

### See also

Description of RedAn02 (Page 2352)
RedAn02 modes (Page 2353)
RedAn02 functions (Page 2353)
RedAn02 error handling (Page 2354)
RedAn02 block diagram (Page 2356)
RedAn02 I/Os (Page 2354)

## 14.6.6 RedAn02 I/Os

### RedAn02 I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In1 | Analog input value 1 | STRUCT<br>• Value: REAL<br>• ST:BYTE | -<br>• 0.0<br>• 16#80 |
| In2 | Analog input value 2 | STRUCT<br>• Value: REAL<br>• ST:BYTE | -<br>• 0.0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| LossRed | 1= Redundancy loss at one of the inputs | STRUCT<br>• Value: BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Out | Output of the process value with the better signal status | STRUCT<br>• Value: REAL<br>• ST:BYTE | -<br>• 0.0<br>• 16#80 |
| SimAct | 1 = one input value has the Simulation status | STRUCT<br>• Value: BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Uncertain | 1 = one input value has the "uncertain" status | STRUCT<br>• Value: BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |

### See also

Description of RedAn02 (Page 2352)
RedAn02 modes (Page 2353)
RedAn02 functions (Page 2353)
RedAn02 error handling (Page 2354)
RedAn02 messaging (Page 2354)
RedAn02 block diagram (Page 2356)

## 14.6.7 RedAn02 block diagram

### RedAn02 block diagram

A block diagram is not provided for this block.

### See also

Description of RedAn02 (Page 2352)
RedAn02 modes (Page 2353)
RedAn02 functions (Page 2353)
RedAn02 messaging (Page 2354)
RedAn02 I/Os (Page 2354)
RedAn02 error handling (Page 2354)

# 14.7 SelA02In - Output of two analog values

## 14.7.1 Description of SelA02In

### Object name (type + number)

Type + number: FB 1886

Family: LogicAn

### Area of application for SelA02In

The block is used for the following applications:

- Selecting one of two analog values and switching it through to the output.

### How it works

Depending on the setting of parameter `SelMode` the block selects one of the two input parameters `In1` or `In2` and writes its value to the output parameter `Out`.

This is displayed at the `In2Selected` output parameter.

### Configuration

Use the CFC editor to install the block in any OB.

For the SelA02In block, the Advanced Process Library contains templates for process tag types as examples and there is a example project (APL_Example_xx, xx designates the language variant) with an application scenario for this block, which describes how the block works.

Examples of process tag types:

- Source chart for GainSched function block (gain scheduling) (Page 2845)
- Override control (Page 2846)
- Override control with PIDConR (OverrideR) (Page 2848)

Application scenario in the example project:

- Process simulation including noise generator (ProcSimC; ProcSimS) (Page 2871)

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

**See also**

SelA02In block diagram (Page 2361)
SelA02In I/Os (Page 2360)
SelA02In messaging (Page 2360)
SelA02In error handling (Page 2359)
SelA02In functions (Page 2358)
SelA02In modes (Page 2358)

## 14.7.2 SelA02In modes

**SelA02In modes**

This block does not have any modes.

**See also**

SelA02In block diagram (Page 2361)
SelA02In I/Os (Page 2360)
SelA02In messaging (Page 2360)
SelA02In error handling (Page 2359)
SelA02In functions (Page 2358)
Description of SelA02In (Page 2357)

## 14.7.3 SelA02In functions

**Functions of SelA02In**

The functions for this block are listed below.

### Select input parameter

The SelMode parameter can affect the selection as follows:

- SelMode ≤ 0: The selection depends on the parameter Sel_In2
  - Sel_In2 = 0: Input parameter In1 is written with its signal status to output parameter Out.
  - Sel_In2 = 1: Input parameter In2 is written with its signal status to output parameter Out.
- SelMode = 1: The input parameter with the lower value (In1 or In2) is written with its signal status to the output parameter Out.
- SelMode ≥ 2: The input parameter with the higher value (In1 or In2) is written with its signal status to the output parameter Out.

The signal status of Sel_In2 is output at the In2Selected output parameter.

### See also

SelA02In block diagram (Page 2361)
SelA02In I/Os (Page 2360)
SelA02In messaging (Page 2360)
SelA02In error handling (Page 2359)
SelA02In modes (Page 2358)
Description of SelA02In (Page 2357)

## 14.7.4 SelA02In error handling

### SelA02In error handling

The block does not report any errors.

### See also

SelA02In block diagram (Page 2361)
SelA02In I/Os (Page 2360)
SelA02In messaging (Page 2360)
SelA02In functions (Page 2358)
SelA02In modes (Page 2358)
Description of SelA02In (Page 2357)

## 14.7.5 SelA02In messaging

### Messaging

This block does not offer messaging.

### See also

SelA02In block diagram (Page 2361)
SelA02In I/Os (Page 2360)
SelA02In error handling (Page 2359)
SelA02In functions (Page 2358)
SelA02In modes (Page 2358)
Description of SelA02In (Page 2357)

## 14.7.6 SelA02In I/Os

### SelA02In I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `EN` | 1 = Called block will be processed | BOOL | 1 |
| `In1` | Analog process value 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `In2` | Analog process value 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| `SelMode` | Selection of the function:<br>0 = Select with `Sel_In2`<br>1 = Minimum<br>2 = Maximum | INT | 0 |
| `Sel_In2` | Selecting the input parameter:<br>0 = `In1`<br>1 = `In2` | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| In2Selected | Selected input parameter:<br>0 = In1<br>1 = In2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Out | Analog process value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### See also

SelA02In block diagram (Page 2361)
SelA02In messaging (Page 2360)
SelA02In error handling (Page 2359)
SelA02In functions (Page 2358)
SelA02In modes (Page 2358)
Description of SelA02In (Page 2357)

## 14.7.7 SelA02In block diagram

### SelA02In block diagram

A block diagram is not provided for this block.

### See also

SelA02In I/Os (Page 2360)
SelA02In messaging (Page 2360)
SelA02In error handling (Page 2359)
SelA02In functions (Page 2358)
SelA02In modes (Page 2358)
Description of SelA02In (Page 2357)

# 14.8 SelA16In - Output of 16 analog values

## 14.8.1 Description of SelA16In

### Object name (type + number) and family

Type + number: FB 1888

Family: LogicAn

### Area of application for SelA16In

The block is used for the following applications:

- Selecting one of 16 analog values and switching it through to the output.

### How it works

The block writes the value of one of the input parameters `In01` through `In16` to the output parameter `Out`. The selection is performed via the `SelInt` input parameter.

The signal status of the selected input parameter is written to the signal status of the output parameter `Out`. There is no additional signal status evaluation.

The unit `InxUnit` of the selected input parameter `Inx` (x = 01 ...16) is written to the output parameter `OutUnit`.

The analog input 1 (`In01`) is operable only if `OS_Perm.Bit22` is set to 1.

It has an upper limit associated with the parameter `In01HiLim` and a lower limit associated with the parameter `In01LoLim`.

If `Feature.Bit7` is set to 1 and the permission is enabled (`OS_Perm.Bit22` = 1), the analog input 1 (`In01`) can be reserved for the operator. If `Feature.Bit7` is set to 1, the analog input 1 (`In01`) is always visible independent of `Feature.Bit5`.

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

You can find a description for each parameter in section SelA16In I/Os (Page 2368).

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | Not used |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | Not used |
| 6 | `OnAct.Value` |
| 7 - 11 | Not used |
| 12 – 27 | Used to highlight the green line in the standard view |
| 28 | 1 = With internal/external selection from the operator (`Feature.Bit6`) |
| 29 | 1 = `SelExtAct` |
| 30 | Not used |
| 31 | Internal error |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | 1 = Display input 1 |
| ... | ... |
| 15 | 1 = Display input 16 |
| 16 - 31 | Not used |

### See also

SelA16In block diagram (Page 2372)

SelA16In messaging (Page 2368)

SelA16In error handling (Page 2367)

SelA16In functions (Page 2364)

SelA16In modes (Page 2363)

## 14.8.2 SelA16In modes

### SelA16In operating modes

The block can be operated using the following modes:

- On (Page 77)
- Out of service (Page 77)

### "On"

You can find general information about the "On" mode in the section On (Page 77).

### "Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

When the mode is changed to "Out of service", the status of the following output parameters with `Feature.Bit3` =1 and `Feature.Bit3` =0 is as shown below:

| Output Parameters | **`Feature.Bit3` =0** | **`Feature.Bit3` =1** |
|---|---|---|
| Out.ST | 16#XX | 16#60 |

### See also

SelA16In block diagram (Page 2372)

SelA16In I/Os (Page 2368)

SelA16In messaging (Page 2368)

SelA16In error handling (Page 2367)

SelA16In functions (Page 2364)

Description of SelA16In (Page 2362)

## 14.8.3 SelA16In functions

### Functions of SelA16In

The functions for this block are listed below.

### Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

### Operator control permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 =Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 =Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can set the input 1 |
| 5 | 1 = Operator can set the input 2 |
| 6 | 1 = Operator can set the input 3 |
| 7 | 1 = Operator can set the input 4 |
| 8 | 1 = Operator can set the input 5 |
| 9 | 1 = Operator can set the input 6 |
| 10 | 1 = Operator can set the input 7 |
| 11 | 1 = Operator can set the input 8 |
| 12 | 1 = Operator can set the input 9 |
| 13 | 1 = Operator can set the input 10 |
| 14 | 1 = Operator can set the input 11 |
| 15 | 1 = Operator can set the input 12 |
| 16 | 1 = Operator can set the input 13 |
| 17 | 1 = Operator can set the input 14 |
| 18 | 1 = Operator can set the input 15 |
| 19 | 1 = Operator can set the input 16 |
| 20 | 1 = Operator can switch to external selection |
| 21 | 1 = Operator can switch to internal selection |
| 22 | 1 = Operator can change the value of input 1 |
| 23 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

## Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status of analog logic blocks (Page 120).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `In1.ST`

etc. to

- `In16.ST`

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) section. The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 3 | Changes signal status of outputs in OOS (Page 208) |
| 4 | Setting switch or button mode (Page 177) |
| 5 | Display only input values that are interconnected in the faceplate (Page 168) |
| 6 | External/internal selection specification (Page 194) |
| 7 | Analog input 1 is reserved for the operator (Page 197) |
| 24 | Enabling local operator authorization (Page 169) |

### Cascading of SelA16In

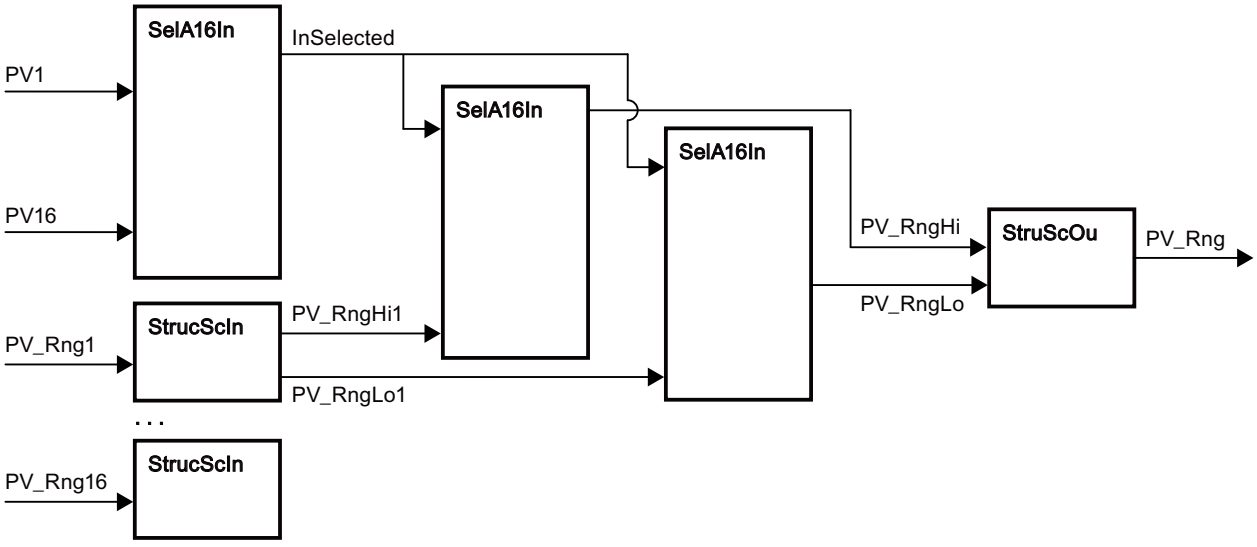



Figure 14-1 Process value area can be realized using conversion blocks and cascaded SelA16In blocks.

### SIMATIC BATCH functionality

This block provides the standard function Auto-Hotspot (Page 73).

### See also

SelA16In block diagram (Page 2372)

SelA16In I/Os (Page 2368)

SelA16In messaging (Page 2368)

SelA16In error handling (Page 2367)

SelA16In modes (Page 2363)

Description of SelA16In (Page 2362)

## 14.8.4 SelA16In error handling

### Error handling of SelA16In

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 18 | 1 > SelExt > 16 |

### See also

SelA16In block diagram (Page 2372)

SelA16In I/Os (Page 2368)

SelA16In messaging (Page 2368)

SelA16In functions (Page 2364)

SelA16In modes (Page 2363)

Description of SelA16In (Page 2362)

## 14.8.5 SelA16In messaging

### Messaging

This block does not offer messaging.

### See also

SelA16In block diagram (Page 2372)

SelA16In I/Os (Page 2368)

SelA16In error handling (Page 2367)

SelA16In functions (Page 2364)

SelA16In modes (Page 2363)

Description of SelA16In (Page 2362)

## 14.8.6 SelA16In I/Os

### I/Os of SelA16In

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| BatchEn | 1 = Enable allocation | BOOL | 0 |
| BatchID | Batch ID | DWORD | 16#00000000 |
| BatchName | Batch name | S7-String | |
| BlkTypeInfo* | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| EN | 1 = Called block will be processed | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| `Feature` | I/O for additional functions (Page 2364) | STRUCT | - |
| | | • Bit 0: BOOL | • 0 |
| | | • ... | • 0 |
| | | • Bit 31: BOOL | • 0 |
| `In01*` | Analog input parameter 1 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In01HiLim` | High limit for the parameter `In01` | REAL | 100.0 |
| `In01LoLim` | Low limit for the parameter `In01` | REAL | 0.0 |
| `In02` | Analog input parameter 2 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In03` | Analog input parameter 3 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In04` | Analog input parameter 4 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In05` | Analog input parameter 5 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In06` | Analog input parameter 6 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In07` | Analog input parameter 7 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In08` | Analog input parameter 8 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In09` | Analog input parameter 9 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In10` | Analog input parameter 10 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In11` | Analog input parameter 11 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| In12 | Analog input parameter 12 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| In13 | Analog input parameter 13 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| In14 | Analog input parameter 14 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| In15 | Analog input parameter 15 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| In16 | Analog input parameter 16 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| In01Unit | Unit for parameter In01 | INT | 1001 |
| In02Unit | Unit for parameter In02 | INT | 1001 |
| In03Unit | Unit for parameter In03 | INT | 1001 |
| In04Unit | Unit for parameter In04 | INT | 1001 |
| In05Unit | Unit for parameter In05 | INT | 1001 |
| In06Unit | Unit for parameter In06 | INT | 1001 |
| In07Unit | Unit for parameter In07 | INT | 1001 |
| In08Unit | Unit for parameter In08 | INT | 1001 |
| In09Unit | Unit for parameter In09 | INT | 1001 |
| In10Unit | Unit for parameter In10 | INT | 1001 |
| In11Unit | Unit for parameter In11 | INT | 1001 |
| In12Unit | Unit for parameter In12 | INT | 1001 |
| In13Unit | Unit for parameter In13 | INT | 1001 |
| In14Unit | Unit for parameter In14 | INT | 1001 |
| In15Unit | Unit for parameter In15 | INT | 1001 |
| In16Unit | Unit for parameter In16 | INT | 1001 |
| LiOp | • With Feature.Bit6 = 0:<br>Selection of the input parameter via:<br>– 0 = Operator (SelInt)<br>– 1 = Interconnection or SFC (SelExt)<br>• With Feature.Bit6 = 1:<br>Internal/external selection via:<br>– 0 = Operator (SelIntOp or SelExtOp)<br>– 1 = Interconnection or SFC (SelIntLi or SelExtLi) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operating permission, connected with the Out output parameter of the upstream block, OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator control permissions (Page 2364) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| SelExt | External selection of the input parameter:<br>1 = In01 selected<br>16 = In16 selected<br>Selection outside the range 1...16 is not possible | INT | 1 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | |
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | |
| SelInt* | Selecting the input parameter:<br>1 = In01 selected<br>16 = In16 selected<br>You cannot select anything outside the range of 1 to 16. | INT | 1 |
| SelExtLi | Linkable input for external selection:<br>1 = External selection<br>Active only with Feature.Bit6 = 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SelExtOp | Operator input for external selection:<br>1 = External selection<br>Active only with Feature.Bit6 = 1 | BOOL | 0 |
| SelIntLi | Linkable input for internal selection:<br>1 = Internal selection<br>Active only with Feature.Bit6 = 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SelIntOp | Operator Input for Internal Selection:<br>1 = Internal selection<br>Active only with Feature.Bit6 = 1 | BOOL | 0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see SelA16In error handling (Page 2367). | INT | -1 |
| InSelected | Selected input parameter:<br>1 = In01 selected<br>16 = In16 selected | STRUCT<br>• Value: INT<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| Out | Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| OutUnit | Unit for parameter Out | INT | 1001 |
| SelExtAct | 0 = Internal selection is active<br>1 = External selection is active | BOOL | 0 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status | Status word (Page 2362) | DWORD | 16#00000000 |
| Status2 | Status word (Page 2362) | DWORD | 16#00000000 |

### See also

SelA16In block diagram (Page 2372)

SelA16In messaging (Page 2368)

SelA16In modes (Page 2363)

## 14.8.7 SelA16In block diagram

### SelA16In block diagram

A block diagram is not provided for this block.

**See also**

SelA16In I/Os (Page 2368)
SelA16In messaging (Page 2368)
SelA16In error handling (Page 2367)
SelA16In functions (Page 2364)
SelA16In modes (Page 2363)
Description of SelA16In (Page 2362)

## 14.8.8 Operator control and monitoring

### 14.8.8.1 SelA16In views

**Views of the SelA16In block**

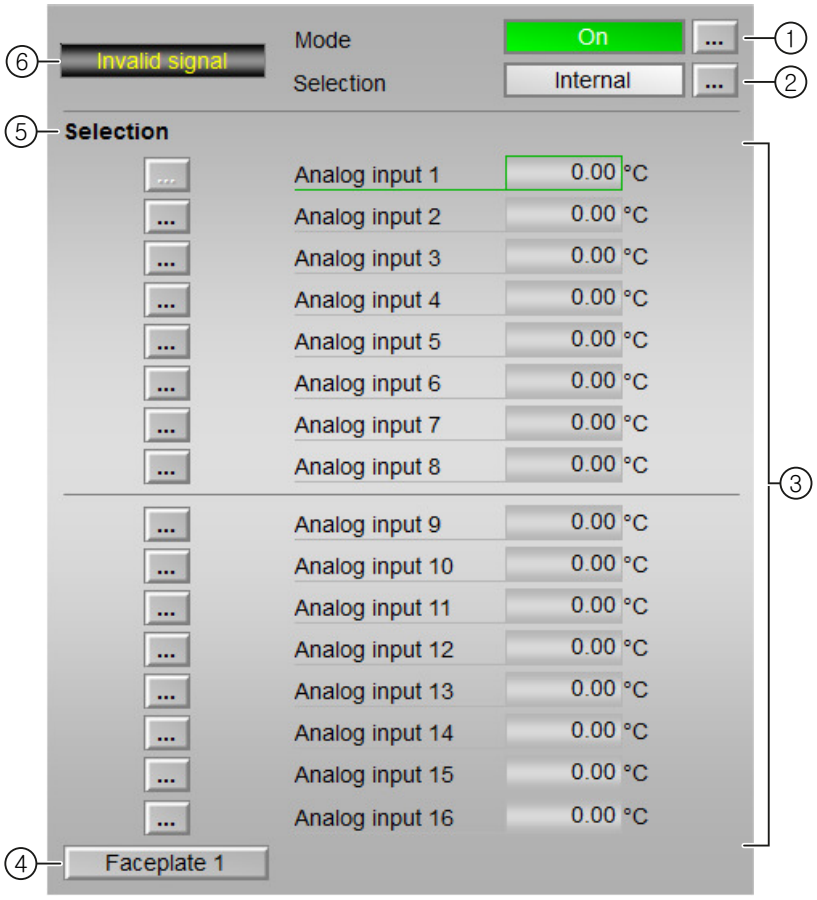
The block SelA16In provides the following views:

- SelA16In standard view (Page 2374)
- SelA16In preview (Page 2376)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for SelA16In (Page 2377)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 14.8.8.2 SelA16In standard view

#### SelA16In standard view




#### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 77)
- Out of service (Page 77)

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the operating mode.

#### (2) Displaying and switching the selection

This area provides information on the current selection. The following selection can be made:

- External: Selecting values externally (program)
- Internal: Selecting values internally (operator)

### (3) Display of analog inputs

This area shows you the analog values connected in the ES for this block.

You can find additional information on this in Switching operating states and operating modes (Page 289).

### (4) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Opening additional faceplates (Page 234) section.

### (5) Selection of analog inputs

You can select the required analog input by clicking on the [...] button.

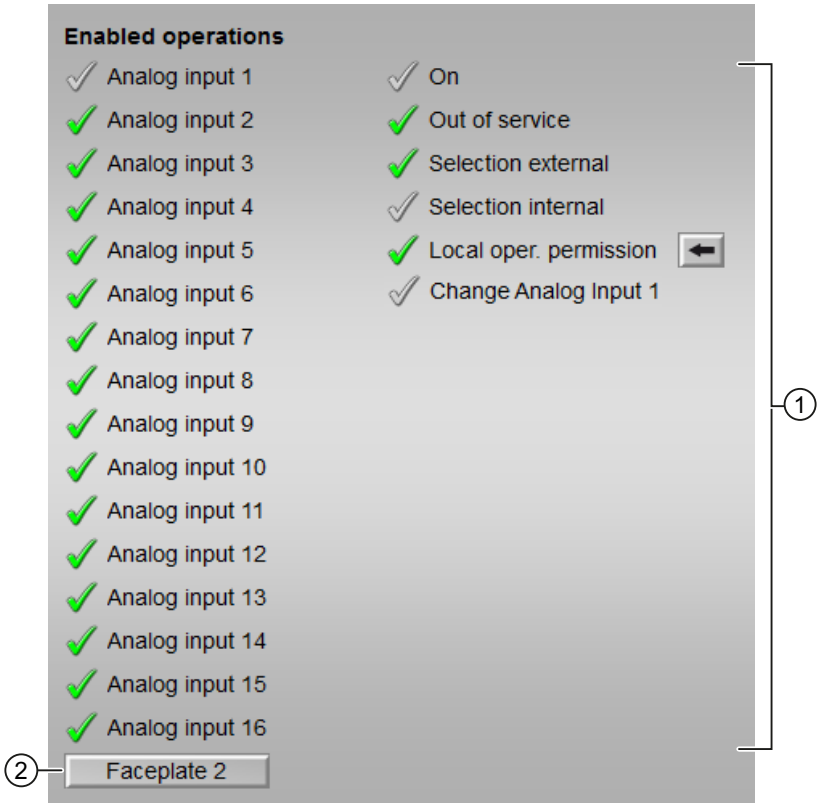
### (6) Display area for block states

This area provides additional information on the operating state of the block:

- Internal error
- Invalid Signal

### 14.8.8.3 SelA16In preview

**Preview of SelA16In**




**(1) Enabled operations**

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- Analog input 1 to 16: You can switch to this analog input.
- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Selection external": You can select values externally (program).

- "Selection internal": You can select values internally (operator).
- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. Additional information is available in the section Operator control permissions (Page 285).
- "Change analog input 1": You can change the value of analog input 1.

### (2) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Opening additional faceplates (Page 234) section.

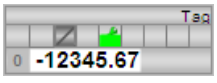
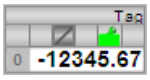
## 14.8.8.4 Block icon for SelA16In

### Block icons for SelA16In

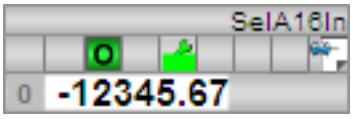
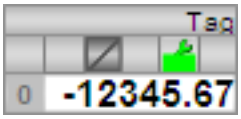
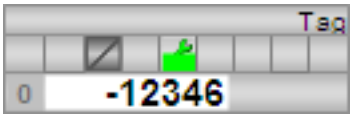
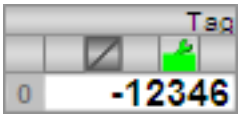
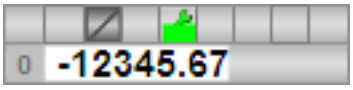
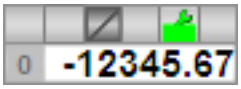
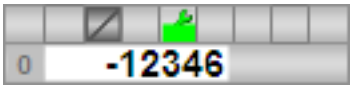
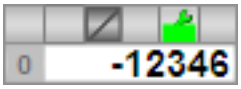
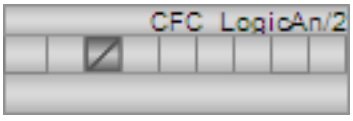
A variety of block icons are available with the following functions:

- Process tag type
- Operating modes
- Signal status, release for maintenance
- Memo display
- Display of the selected analog value

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | Block icon in the full display |

The block icons from template @TemplateAPLV7.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | Block icon in the full display |
|  | 2 | |
|  | 3 | |
|  | 4 | |
|  | 5 | |
|  | 6 | |
|  | 7 | |
|  | 8 | |
|  | - | Block icon in "Out of service" mode (example with type 1 block icon) |

**Note**

The display of the unit in the types 1/3/5/7 depends on the analog value used, although the unit can only be displayed correctly when the units of the inputs have been configured with a code (see "Selecting a unit of measure (Page 240)"). Units configured with the S7_unit attribute are not supported.

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 267)
- Block icon structure (Page 260)
- Operation via the block icon (Page 269).

# 14.9 SelAn16L - Large version of SelAn16In

## 14.9.1 Description of SelAn16L

### Object name (type + number) and family

Type + number: FB 1889

Family: LogicAn

### Area of application for SelAn16L

The block is used for the following applications:

1. Selecting one of 16 analog values and switching it through to the output.
2. Find maximum and minimum value among all connected inputs.

### How it works

The block writes the value of one of the input parameters `In01` through `In16` to the output parameter `Out`. The selection is performed via the `SelInt` input parameter.

The signal status of the selected input parameter is written to the signal status of the output parameter `Out`. There is no additional signal status evaluation.

The unit `InxUnit` of the selected input parameter `Inx` (x = 01 ...16) is written to the output parameter `OutUnit`.

The analog input 1 (`In01`) is operable only if `OS_Perm.Bit22` is set to 1.

It has an upper limit associated with the parameter `In01HiLim` and a lower limit associated with the parameter `In01LoLim`.

If `Feature.Bit7` is set to 1 and the permission is enabled (`OS_Perm.Bit22` = 1), the analog input 1 (`In01`) can be reserved for the operator. If `Feature.Bit7` is set to 1, the analog input 1 (`In01`) is always visible independent of `Feature.Bit5`.

This block also provides the following:

- Maximum and minimum values of connected inputs only along with its units.
- OpScale for the output `Out`.

**Note**

In case no inputs are connected, then the output of Maximum and Minimum value will be 0 and the signal status will be 16#FF.

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

You can find a description for each parameter in section SelAn16L I/Os (Page 2386) .

| Status bit | Parameter |
|---|---|
| 0 | `Occupied` |
| 1 | `BatchEn` |
| 2 | Not used |
| 3 | `OosAct.Value` |
| 4 | `OosLi.Value` |
| 5 | Not used |
| 6 | `OnAct.Value` |
| 7 - 11 | Not used |
| 12 – 27 | Used to highlight the green line in the standard view |
| 28 | 1 = With internal/external selection from the operator (`Feature.Bit6`) |
| 29 | 1 = `SelExtAct` |
| 30 | Not used |
| 31 | Internal error |

### Status word allocation for `Status2` parameter

| Status bit | Parameter |
|---|---|
| 0 | 1 = Display input 1 |
| ... | ... |
| 15 | 1 = Display input 16 |
| 16 - 31 | Not used |

### See also

SelAn16L modes (Page 2382)

SelAn16L messaging (Page 2386)

SelAn16L error handling (Page 2385)

SelAn16L functions (Page 2382)

SelAn16L block diagram (Page 2392)

## 14.9.2 SelAn16L modes

### SelAn16L operating modes

The block can be operated using the following modes:

- On (Page 77)
- Out of service (Page 77)

### On

You can find general information about the "On" mode in the section On (Page 77).

### Out of service

You can find general information about the "Out of service" mode in the section Out of service (Page 77).

With `Feature.Bit3` =1 the following output parameter become the status of simulation 16#60, if mode is changed to "Out of service".

With `Feature.Bit3` =0 the following output parameter become the status of 16#80, if mode is changed to "Out of service".

- Out.ST
- In_Min.ST
- In_Max.ST

### See also

Description of SelAn16L (Page 2380)

SelAn16L functions (Page 2382)

SelAn16L error handling (Page 2385)

SelAn16L messaging (Page 2386)

SelAn16L block diagram (Page 2392)

SelAn16L I/Os (Page 2386)

## 14.9.3 SelAn16L functions

### Functions of SelAn16L

The functions for this block are listed below.

## Opening additional faceplates

This block provides the standard function Opening additional faceplates (Page 234).

## Operator control permissions

This block provides the standard function Operator control permissions (Page 285).

The block has the following permissions for the `OS_Perm` parameter:

| Bit | Function |
|---|---|
| 0 | Not used |
| 1 | 1 =Operator can switch to "On" mode |
| 2 | Not used |
| 3 | 1 =Operator can switch to "Out of service" mode |
| 4 | 1 = Operator can set the input 1 |
| 5 | 1 = Operator can set the input 2 |
| 6 | 1 = Operator can set the input 3 |
| 7 | 1 = Operator can set the input 4 |
| 8 | 1 = Operator can set the input 5 |
| 9 | 1 = Operator can set the input 6 |
| 10 | 1 = Operator can set the input 7 |
| 11 | 1 = Operator can set the input 8 |
| 12 | 1 = Operator can set the input 9 |
| 13 | 1 = Operator can set the input 10 |
| 14 | 1 = Operator can set the input 11 |
| 15 | 1 = Operator can set the input 12 |
| 16 | 1 = Operator can set the input 13 |
| 17 | 1 = Operator can set the input 14 |
| 18 | 1 = Operator can set the input 15 |
| 19 | 1 = Operator can set the input 16 |
| 20 | 1 = Operator can switch to external selection |
| 21 | 1 = Operator can switch to internal selection |
| 22 | 1 = Operator can change the value of input 1 |
| 23 - 31 | Not used |

**Note**

If you interconnect a parameter that is also listed in `OS_Perm` as a parameter, you have to reset the corresponding `OS_Perm` bit.

## Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status of analog logic blocks (Page 120).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `In1.ST`

etc. to

- `In16.ST`

### Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 240).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) section. The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 1 | Reaction to the out of service mode (Page 188) |
| 2 | Resetting the commands for changing the mode (Page 172) |
| 3 | Changes signal status of outputs in OOS (Page 208) |
| 4 | Setting switch or button mode (Page 177) |
| 5 | Display only input values that are interconnected in the faceplate (Page 168) |
| 6 | External/internal selection specification (Page 194) |
| 7 | Analog input 1 is reserved for the operator (Page 197) |
| 24 | Enabling local operator authorization (Page 169) |

### Cascading of SelA16In

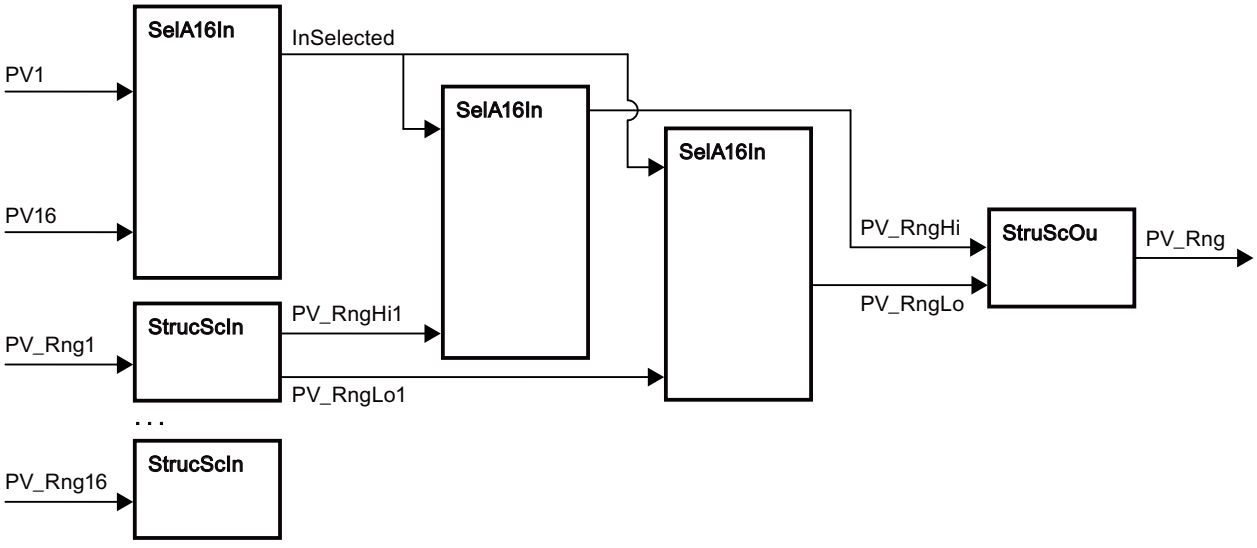



Figure 14-2 Process value area can be realized using conversion blocks and cascaded SelA16In blocks.

### SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 73).

### See also

SelAn16L block diagram (Page 2392)

Description of SelAn16L (Page 2380)

SelAn16L modes (Page 2382)

SelAn16L error handling (Page 2385)

SelAn16L messaging (Page 2386)

SelAn16L I/Os (Page 2386)

## 14.9.4 SelAn16L error handling

### Error handling of SelAn16L

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 18 | 1 > SelExt > 16 |

### See also

Description of SelAn16L (Page 2380)

SelAn16L modes (Page 2382)

SelAn16L functions (Page 2382)

SelAn16L messaging (Page 2386)

SelAn16L I/Os (Page 2386)

SelAn16L block diagram (Page 2392)

## 14.9.5 SelAn16L messaging

### Messaging

This block does not offer messaging.

### See also

Description of SelAn16L (Page 2380)

SelAn16L modes (Page 2382)

SelAn16L functions (Page 2382)

SelAn16L error handling (Page 2385)

SelAn16L I/Os (Page 2386)

SelAn16L block diagram (Page 2392)

## 14.9.6 SelAn16L I/Os

### I/Os of SelAn16L

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `BatchEn` | 1 = Enable allocation | BOOL | 0 |
| `BatchID` | Batch ID | DWORD | 16#00000000 |
| `BatchName` | Batch name | S7-String | |
| `BlkTypeInfo*` | Block Type Version and FB Number of blocktype<br>The block version is coded as a hex value and includes the major, minor, SP and update version of the block type.<br>VersId=16#wxyz<br>w = major version<br>x = minor version<br>y = service pack<br>z = update<br>This input can not be edit or linked and is for internal usage only | STRUCT<br>• TypNr : INT<br>• VersId : WORD | FB-Number<br>16#9100 |
| `EN` | 1 = Called block will be processed | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| `Feature` | I/O for additional | STRUCT | - |
| | | • Bit 0: BOOL | • 0 |
| | | • ... | • 0 |
| | | • Bit 31: BOOL | • 0 |
| `In01*` | Analog input parameter 1 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In01HiLim` | High limit for the parameter `In01` | REAL | 100.0 |
| `In01LoLim` | Low limit for the parameter `In01` | REAL | 0.0 |
| `In02` | Analog input parameter 2 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In03` | Analog input parameter 3 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In04` | Analog input parameter 4 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In05` | Analog input parameter 5 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In06` | Analog input parameter 6 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In07` | Analog input parameter 7 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In08` | Analog input parameter 8 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In09` | Analog input parameter 9 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In10` | Analog input parameter 10 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |
| `In11` | Analog input parameter 11 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#FF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| In12 | Analog input parameter 12 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| In13 | Analog input parameter 13 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| In14 | Analog input parameter 14 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| In15 | Analog input parameter 15 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| In16 | Analog input parameter 16 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#FF |
| In01Unit | Unit for parameter In01 | INT | 1001 |
| In02Unit | Unit for parameter In02 | INT | 1001 |
| In03Unit | Unit for parameter In03 | INT | 1001 |
| In04Unit | Unit for parameter In04 | INT | 1001 |
| In05Unit | Unit for parameter In05 | INT | 1001 |
| In06Unit | Unit for parameter In06 | INT | 1001 |
| In07Unit | Unit for parameter In07 | INT | 1001 |
| In08Unit | Unit for parameter In08 | INT | 1001 |
| In09Unit | Unit for parameter In09 | INT | 1001 |
| In10Unit | Unit for parameter In10 | INT | 1001 |
| In11Unit | Unit for parameter In11 | INT | 1001 |
| In12Unit | Unit for parameter In12 | INT | 1001 |
| In13Unit | Unit for parameter In13 | INT | 1001 |
| In14Unit | Unit for parameter In14 | INT | 1001 |
| In15Unit | Unit for parameter In15 | INT | 1001 |
| In16Unit | Unit for parameter In16 | INT | 1001 |
| In01_OpScale | OS display range for In01 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100<br>• 0 |
| In02_OpScale | OS display range for In02 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100<br>• 0 |
| In03_OpScale | OS display range for In03 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| In04_OpScale | OS display range for In04 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100<br>• 0 |
| In05_OpScale | OS display range for In05 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100<br>• 0 |
| In06_OpScale | OS display range for In06 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100<br>• 0 |
| In07_OpScale | OS display range for In07 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100<br>• 0 |
| In08_OpScale | OS display range for In08 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100<br>• 0 |
| In09_OpScale | OS display range for In09 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100<br>• 0 |
| In10_OpScale | OS display range for In10 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100<br>• 0 |
| In11_OpScale | OS display range for In11 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100<br>• 0 |
| In12_OpScale | OS display range for In12 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100<br>• 0 |
| In13_OpScale | OS display range for In13 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100<br>• 0 |
| In14_OpScale | OS display range for In14 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100<br>• 0 |
| In15_OpScale | OS display range for In15 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100<br>• 0 |
| In16_OpScale | OS display range for In16 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100<br>• 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| LiOp | • With Feature.Bit6 = 0:<br>Selection of the input parameter via:<br>– 0 = Operator (SelInt)<br>– 1 = Interconnection or SFC (SelExt)<br>• With Feature.Bit6 = 1:<br>Internal/external selection via:<br>– 0 = Operator (SelIntOp or SelExtOp)<br>– 1 = Interconnection or SFC (SelIntLi or SelExtLi) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Occupied | 1 = Occupied by batch control | BOOL | 0 |
| OnOp* | 1 = "On" mode via operator | BOOL | 0 |
| OosLi | 1 = "Out of service", via interconnection or SFC (0-1 edge transition) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosOp* | 1 = "Out of service", via OS operator | BOOL | 0 |
| OpSt_In | Input parameter for local operating permission, connected with the Out output parameter of the upstream block, Description of OpStations (Page 457) | DWORD | 16#00000000 |
| OS_Perm | I/O for operator control permissions (Page 460) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 1<br>• 1<br>• 1 |
| SelExt | External selection of the input parameter:<br>1 = In01 selected<br>16 = In16 selected<br>Selection outside the range 1...16 is not possible | INT | 1 |
| SelFp1 | Call a block saved in this parameter as an additional faceplate (Page 234) in the standard view | ANY | |
| SelFp2 | Call a block saved in this parameter as an additional faceplate (Page 234) in the preview | ANY | |
| SelInt* | Selecting the input parameter:<br>1 = In01 selected<br>16 = In16 selected<br>You cannot select anything outside the range of 1 to 16. | INT | 1 |
| SelExtLi | Linkable input for external selection:<br>1 = External selection<br>Active only with Feature.Bit6 = 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SelExtOp | Operator input for external selection:<br>1 = External selection<br>Active only with Feature.Bit6 = 1 | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SelIntLi | Linkable input for internal selection:<br>1 = Internal selection<br>Active only with Feature.Bit6 = 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SelIntOp | Operator Input for Internal Selection:<br>1 = Internal selection<br>Active only with Feature.Bit6 = 1 | BOOL | 0 |
| StepNo | Batch step number | DWORD | 16#00000000 |
| UserStatus | First freely assignable bits for use in OS | BYTE | 16#00 |
| UserStatus 2 | Second freely assignable bits for use in OS | DWORD | 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of current error number. For error numbers that can be output by this block, see SelAn16L error handling (Page 2385). | INT | -1 |
| InSelected | Selected input parameter:<br>1 = In01 selected<br>16 = In16 selected | STRUCT<br>• Value: INT<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| In_Min | Minimum of all connected inputs | STRUCT<br>• Value: INT<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| In_MinUnit | Engineering unit of In_Min | • INT | 1001 |
| In_Max | Maximum of all connected inputs | STRUCT<br>• Value: INT<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| In_MaxUnit | Engineering unit of In_Max | • INT | 1001 |
| OnAct | 1 = "On" mode enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| OosAct | 1 = Block is "Out of service" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpSt_Out | Value of the OpSt_In input parameter, for feedforwarding to other blocks. Bit 31 of this parameter is used by Feature.Bit24 | DWORD | 16#00000000 |
| OS_PermLog | Display of OS_Perm with settings changed by the block algorithm | DWORD | 16#FFFFFFFF |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OS_PermOut | Display of OS_Perm | DWORD | 16#FFFFFFFF |
| Out | Output | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| OutUnit | Unit for parameter Out | INT | 1001 |
| Out_OpScale | OS display range for Out | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100<br>• 0 |
| SelExtAct | 0 = Internal selection is active<br>1 = External selection is active | BOOL | 0 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| Status | Status word (Page 2380) | DWORD | 16#00000000 |
| Status2 | Status word (Page 2380) | DWORD | 16#00000000 |

**See also**

SelAn16L functions (Page 2382)
Opening additional faceplates (Page 234)
SelAn16L modes (Page 2382)
SelAn16L messaging (Page 2386)
SelAn16L block diagram (Page 2392)

## 14.9.7 SelAn16L block diagram

### SelA16In block diagram

A block diagram is not provided for this block.

**See also**

Description of SelAn16L (Page 2380)
SelAn16L modes (Page 2382)
SelAn16L functions (Page 2382)
SelAn16L error handling (Page 2385)
SelAn16L messaging (Page 2386)
SelAn16L I/Os (Page 2386)

## 14.9.8 Operaror control and monitoring

### 14.9.8.1 SenAn16L views

**Views of the SelAn16L block**

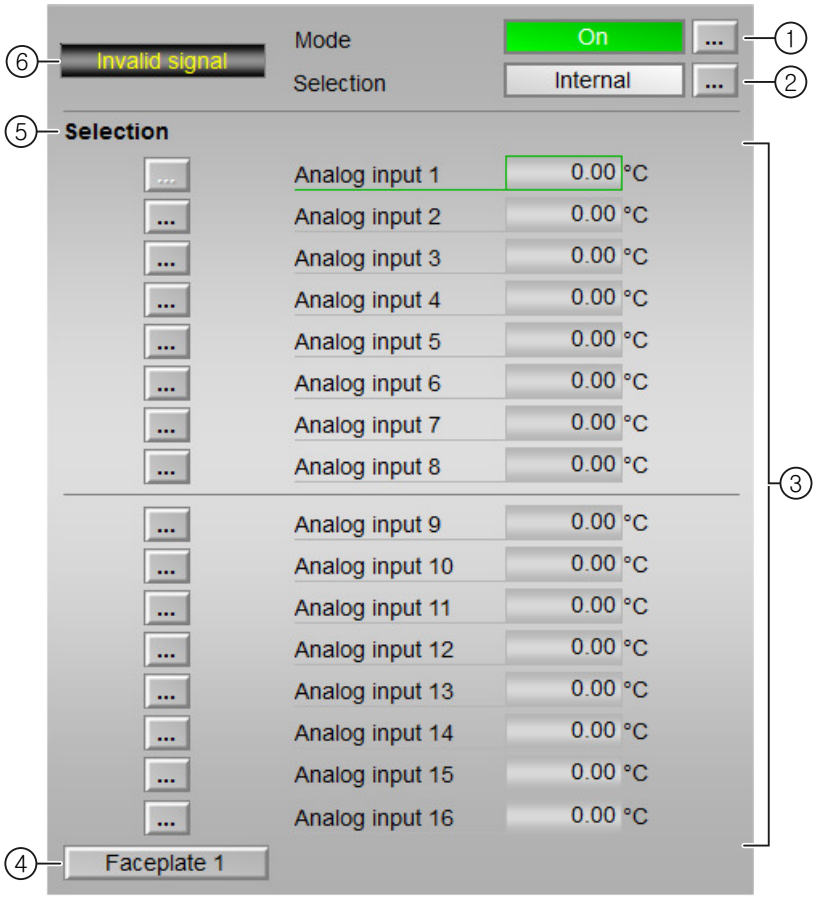
The block SelAn16L provides the following views:

- SelAn16L standard view (Page 2394)
- SelAn16L preview (Page 2395)
- Memo view (Page 338)
- Batch view (Page 336)
- Block icon for SelAn16L (Page 2397)

Refer to the sections Structure of the faceplate (Page 279) and Block icon structure (Page 260) for general information on the faceplate and block icon.

### 14.9.8.2 SelAn16L standard view

### SelAn16L standard view




### (1) Displaying and switching the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 77)
- Out of service (Page 77)

Refer to the Switching operating states and operating modes (Page 289) section for information on switching the operating mode.

### (2) Displaying and switching the selection

This area provides information on the current selection. The following selection can be made:

- External: Selecting values externally (program)
- Internal: Selecting values internally (operator)

### (3) Display of analog inputs

This area shows you the analog values connected in the ES for this block.

You can find additional information on this in Switching operating states and operating modes (Page 289).

### (4) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Opening additional faceplates (Page 234) section.

### (5) Selection of Analog inputs

You can select the required analog input by clicking on the [...] button.

### (6) Display area for block states

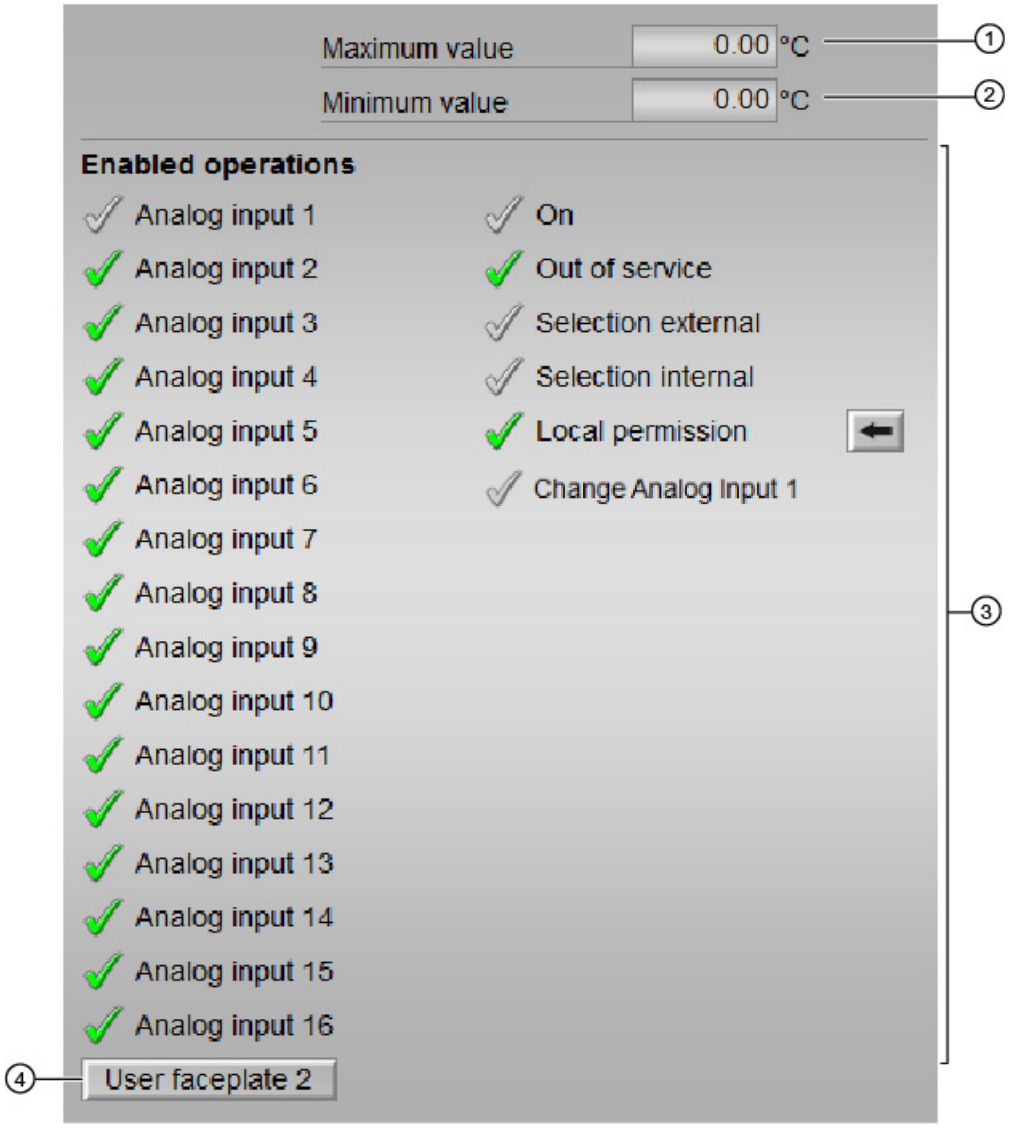
This area provides additional information on the operating state of the block:

- Internal error
- Invalid Signal

## 14.9.8.3 SelAn16L preview

### Preview of SelAn16L

Preview view of the SelAn16L block is as below:




### (1) Maximum Value

This field shows the maximum value among the 16 analog inputs.

### (2) Minimum Value

This field shows the minimum value among the 16 analog inputs.

### (3) Enabled operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for enabled operations:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured AS operator control permissions (`OS_Perm` or `OS1Perm`)

The following enabled operations are shown here:

- Analog input 1 to 16: You can switch to this analog input.
- "On": You can switch to "On" operating mode.
- "Out of service": You can switch to "Out of service" operating mode.
- "Selection external": You can select values externally (program).
- "Selection internal": You can select values internally (operator).
- "Local operating permission": Use the ← button to switch to the standard view of the OpStations block. Additional information is available in the section Operator control permissions.
- "Change analog input 1": You can change the value of analog input 1.

### (4) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Opening additional faceplates section.

**Note**

All other views of Faceplate remain same as that of SelA16In.

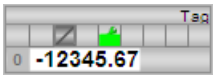
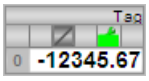
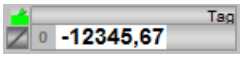
## 14.9.8.4 Block icon for SelAn16L

### Block icons for SelAn16L

A variety of block icons are available with the following functions:

- Process tag type
- Operating modes
- Signal status, release for maintenance
- Memo display
- Display of the selected analog value

The block icons from template @TemplateAPLV9.PDL:

| Icons | Selection of the block icon in CFC | Special features |
|---|---|---|
|  | 1 | |
|  | 2 | |
|  | 3 | Block icon in the full display |

# Digital logic blocks 15

## 15.1 And04 - Forming an AND signal from 4 binary input signals

### 15.1.1 Description of And04

#### Object name (type + number) and family

Type + number: FC 355

Family: LogicDi

#### Area of application for And04

The block is used for the following applications:

- Forming an AND-output signal from four binary input values

#### How it works

Four input parameters are combined via the AND function ("and-ing") into one output value `Out`.

You can use this block to e.g. start or stop a device if all incoming signals are the same.

You can find additional information on forming the signal status under Forming and outputting the signal status of digital logic blocks (Page 119)

#### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

#### Startup characteristics

The block does not have any startup characteristics.

#### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

#### See also

And04 block diagram (Page 2402)

And04 I/Os (Page 2401)

And04 messaging (Page 2401)
And04 error handling (Page 2401)
And04 functions (Page 2400)
And04 modes (Page 2400)

## 15.1.2 And04 modes

### And04 modes

This block does not have any modes.

### See also

And04 block diagram (Page 2402)
And04 I/Os (Page 2401)
And04 messaging (Page 2401)
And04 error handling (Page 2401)
And04 functions (Page 2400)
Description of And04 (Page 2399)

## 15.1.3 And04 functions

### Functions of And04

This block does not have any other functions.

### See also

And04 block diagram (Page 2402)
And04 I/Os (Page 2401)
And04 messaging (Page 2401)
And04 error handling (Page 2401)
And04 modes (Page 2400)
Description of And04 (Page 2399)

## 15.1.4 And04 error handling

### And04 error handling

The block does not report any errors.

### See also

And04 block diagram (Page 2402)
And04 I/Os (Page 2401)
And04 messaging (Page 2401)
And04 functions (Page 2400)
And04 modes (Page 2400)
Description of And04 (Page 2399)

## 15.1.5 And04 messaging

### Messaging

This block does not offer messaging.

### See also

And04 block diagram (Page 2402)
And04 I/Os (Page 2401)
And04 error handling (Page 2401)
And04 functions (Page 2400)
And04 modes (Page 2400)
Description of And04 (Page 2399)

## 15.1.6 And04 I/Os

### And04 I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In1 | Input 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| In2 | Input 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| In3 | Input 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| In4 | Input 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### See also

And04 block diagram (Page 2402)
And04 messaging (Page 2401)
And04 error handling (Page 2401)
And04 functions (Page 2400)
And04 modes (Page 2400)
Description of And04 (Page 2399)

## 15.1.7 And04 block diagram

### And04 block diagram

A block diagram is not provided for this block.

**See also**

And04 I/Os (Page 2401)
And04 messaging (Page 2401)
And04 error handling (Page 2401)
And04 functions (Page 2400)
And04 modes (Page 2400)
Description of And04 (Page 2399)

# 15.2 And08 - Forming an AND signal from 8 binary input signals

## 15.2.1 Description of And08

### Object name (type + number) and family

Type + number: FC 356

Family: LogicDi

### Area of application for And08

The block is used for the following applications:

- Forming an AND- output signal from eight binary input values

### How it works

Eight input parameters are combined via the AND function ("and-ing") into one output value `Out`.

You can use this block to e.g. start or stop a device if all incoming signals are the same.

You can find additional information on forming the signal status under Forming and outputting the signal status of digital logic blocks (Page 119)

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

### See also

And08 block diagram (Page 2408)

And08 I/Os (Page 2406)

And08 messaging (Page 2406)

And08 error handling (Page 2405)

And08 functions (Page 2405)

And08 modes (Page 2405)

## 15.2.2 And08 modes

### And08 modes

This block does not have any modes.

### See also

And08 block diagram (Page 2408)
And08 I/Os (Page 2406)
And08 messaging (Page 2406)
And08 error handling (Page 2405)
And08 functions (Page 2405)
Description of And08 (Page 2404)

## 15.2.3 And08 functions

### Functions of And08

This block does not have any other functions.

### See also

And08 block diagram (Page 2408)
And08 I/Os (Page 2406)
And08 messaging (Page 2406)
And08 error handling (Page 2405)
And08 modes (Page 2405)
Description of And08 (Page 2404)

## 15.2.4 And08 error handling

### And08 error handling

The block does not report any errors.

### See also

And08 block diagram (Page 2408)
And08 I/Os (Page 2406)
And08 messaging (Page 2406)
And08 functions (Page 2405)
And08 modes (Page 2405)
Description of And08 (Page 2404)

## 15.2.5 And08 messaging

### Messaging

This block does not offer messaging.

### See also

And08 block diagram (Page 2408)
And08 I/Os (Page 2406)
And08 error handling (Page 2405)
And08 functions (Page 2405)
And08 modes (Page 2405)
Description of And08 (Page 2404)

## 15.2.6 And08 I/Os

### And08 I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In1 | Input 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| In2 | Input 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| In3 | Input 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| In4 | Input 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| In5 | Input 5 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| In6 | Input 6 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| In7 | Input 7 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| In8 | Input 8 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### See also

And08 block diagram (Page 2408)
And08 messaging (Page 2406)
And08 error handling (Page 2405)
And08 functions (Page 2405)
And08 modes (Page 2405)
Description of And08 (Page 2404)

## 15.2.7 And08 block diagram

### And08 block diagram

A block diagram is not provided for this block.

### See also

And08 I/Os (Page 2406)
And08 messaging (Page 2406)
And08 error handling (Page 2405)
And08 functions (Page 2405)
And08 modes (Page 2405)
Description of And08 (Page 2404)

# 15.3 FlipFlop - Flip Flop with Clock Input

## 15.3.1 Description of FlipFlop

### Object name (type + number) and family

Type + number: FC 389

Family: LogicDi

### Area of application for FlipFlop

The block is used for the following applications:

- SR/RS FlipFlop functionality
- JK-FlipFlop functionality

### How it works

The block provides the function of two types of a flip-flop:

- JK-Flip-Flop (JK_En=1)
- SR/RS-FlipFlop (JK_En=0)

The SR/RS behaviour can be set via the Mode input.

- Mode = 0: SR-Behavior
- Mode = 1: RS-Behavior

If the FlipFlop operates as JK-FlipFlop, the inputs for setting (`SetLi`) and resetting (`RstLi`) are accepted with a positive edge at the input `Clock` and the output `P_Clock` is set for one block cycle.

With a positive edge at the input parameter for setting `SetLi`, the output parameter `Out` is set to 1. Simultaneously, the output parameter `InvOut` is reset.

With a positive edge at the input parameter for resetting `RstLi`, the output parameter `Out` is reset. Simultaneously, the output parameter `InvOut` is set.

If the two input parameters are`SetLi` and `RstLi = 0`, then the block retains its state. In this case, nothing changes at the output parameters.

### How the block operates as SR-FlipFlop (`Mode` = 0) (`JK_En` = 0)

The input parameter for setting `SetLi` has priority over the input parameter for resetting `RstLi`.

Truth table:

| RstLi | SetLi | Out | InvOut |
|---|---|---|---|
| 0 | 0 | Last value | Last value |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

### How the block operates as RS-FlipFlop (Mode = 1) (JK_En = 0)

The input parameter for resetting `RstLi` has priority over the input parameter for setting `SetLi`.

Truth table:

| RstLi | SetLi | Out | InvOut |
|---|---|---|---|
| 0 | 0 | Last value | Last value |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 |

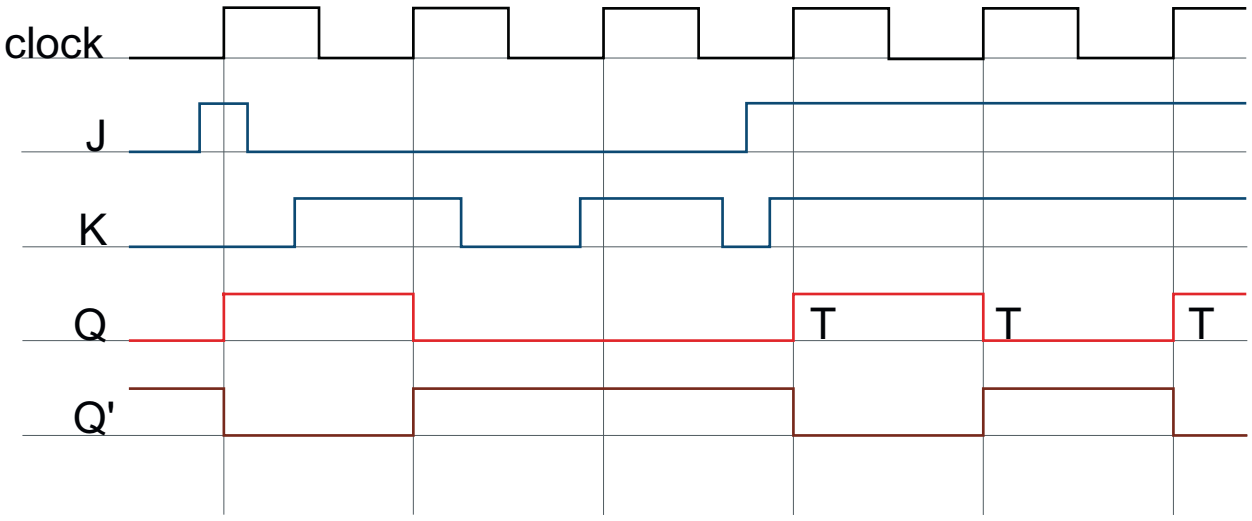
### How the block operates as JK-FlipFlop (JK_En = 1)

The latch works as a JK-latch, the value of the input `Mode` is not relevant in this case.

Truth table:

| Clock | RstLi | SetLi | Out | P_Clock |
|---|---|---|---|---|
| 0 | x | x | unchanged | 0 |
| 1 | x | x | unchanged | 0 |
| 0→1 | 0 | 1 | 1 (set latch) | 1 (one cycle) |
| 0→1 | 1 | 0 | 0 (reset latch) | 1 (one cycle) |
| 0→1 | 1 | 1 | toggle | 1 (one cycle) |
| x = "0" or "1" | | | | |

This timing diagram of JK-latch is as follows:




Clock → `Clock`
J → `SetLi`
K → `RstLi`.
Q → `Out`
Q' → `InvOut`

T → toggle

## Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

## Startup characteristics

The block does not have any startup characteristics.

A startup behavior can be reached with a connection of the input parameter `Init` to the output parameter `InitOut` of the Trigger block. For more information, refer to Startup characteristics over Trigger block (Page 75).

## Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

## See also

FlipFlop modes (Page 2412)

FlipFlop functions (Page 2412)

FlipFlop error handling (Page 2413)

FlipFlop messaging (Page 2414)

FlipFlop I/Os (Page 2414)

FlipFlop block diagram (Page 2416)

## 15.3.2 FlipFlop modes

### FlipFlop operating modes

This block does not have any modes.

### See also

Description of FlipFlop (Page 2409)
FlipFlop functions (Page 2412)
FlipFlop error handling (Page 2413)
FlipFlop messaging (Page 2414)
FlipFlop I/Os (Page 2414)
FlipFlop block diagram (Page 2416)

## 15.3.3 FlipFlop functions

### Functions of FlipFlop

The block provides the following functions:

### Forming the signal status for blocks – SR-FlipFlop

The signal status is formed as follows:

| RstLi | SetLi | Out.ST | InvOut.ST |
|---|---|---|---|
| 0 | 0 | Last signal status | Last signal status |
| 0 | 1 | = SetLi.ST | = SetLi.ST |
| 1 | 0 | = RstLi.ST | = RstLi.ST |
| 1 | 1 | = SetLi.ST | = SetLi.ST |

### Forming the signal status for blocks - RS-FlipFlop

The signal status is formed as follows:

| RstLi | SetLi | Out.ST | InvOut.ST |
|---|---|---|---|
| 0 | 0 | Last signal status | Last signal status |
| 0 | 1 | = SetLi.ST | = SetLi.ST |
| 1 | 0 | = RstLi.ST | = RstLi.ST |
| 1 | 1 | = RstLi.ST | = RstLi.ST |

### Forming the signal status for blocks - JK-FlipFlop

The signal status is formed as follows:

| RstLi | SetLi | Clock | Out.ST |
|---|---|---|---|
| 0 | 0 | 0 | Last signal status |
| 0 | 1 | 0→1 | = SetLi.ST |
| 1 | 0 | 0→1 | = RstLi.ST |
| 1 | 1 | 0→1 | If Out set to "1"<br>= SetLi.ST |
| | | | If Out set to "0"<br>= RstLi.ST |

### Initialization

With the input parameter `Init`, the output parameters `Out` and `InvOut` can be initialized:

- `Init` = 0 (default): Normal flip-flop behavior
  The `Out` and `InvOut` output parameters are calculated according to the mode parameterization and also using the values of the `SetLi` and `RstLi` inputs. For more information see, "How it works" under Description of FlipFlop (Page 2409)
- `Init` = 1: Outputs will be initialized as follow:

| Mode | Out | InvOut | Out.ST and InvOut.ST |
|---|---|---|---|
| 0 | 0 | 1 | RstLi.ST |
| 1 | 1 | 0 | SetLi.ST |

To initialize the output parameters `Out` and `InvOut` after a startup, connect the input parameter `Init` with the output parameter `InitOut` of the Trigger block. For more information, refer to Startup characteristics over Trigger block (Page 75).

### See also

FlipFlop modes (Page 2412)
FlipFlop error handling (Page 2413)
FlipFlop messaging (Page 2414)
FlipFlop I/Os (Page 2414)
FlipFlop block diagram (Page 2416)

## 15.3.4 FlipFlop error handling

### FlipFlop error handling

The block does not report any errors.

### See also

Description of FlipFlop (Page 2409)
FlipFlop modes (Page 2412)
FlipFlop functions (Page 2412)
FlipFlop messaging (Page 2414)
FlipFlop I/Os (Page 2414)
FlipFlop block diagram (Page 2416)

## 15.3.5 FlipFlop messaging

### Messaging

This block does not offer messaging.

### See also

Description of FlipFlop (Page 2409)
FlipFlop modes (Page 2412)
FlipFlop functions (Page 2412)
FlipFlop error handling (Page 2413)
FlipFlop I/Os (Page 2414)
FlipFlop block diagram (Page 2416)

## 15.3.6 FlipFlop I/Os

### FlipFlop I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `Clock` | Linkable clock Signal | BOOL | 0 |
| `EN` | 1 = Called block will be processed | BOOL | 1 |
| `Init` | Initialize function block | BOOL | 0 |
| `JK_En` | 1 = enabeles JK-FlipFlop | BOOL | 0 |
| `Mem` | for internal use | STRUCT | |
| `Mem.mClk` | for internal use | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Mem.mOut | for internal use | BOOL | 0 |
| Mem.m2 | for internal use | BOOL | 0 |
| Mem.m3 | for internal use | BOOL | 0 |
| Mem.m4 | for internal use | BOOL | 0 |
| Mem.m5 | for internal use | BOOL | 0 |
| Mem.m6 | for internal use | BOOL | 0 |
| Mem.m7 | for internal use | BOOL | 0 |
| Mode | Specifying how the block works:<br>• 0 = SR-FlipFlop<br>• 1 = RS-FlipFlop | BOOL | 0 |
| RstLi | 1 = Reset via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SetLi | 1 = Set via interconnection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| InvOut | Inverted output signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| Out | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| P_Clock | pulse output 0→1 edge deteced at Clock input<br>(always "1" if JK_En=0) | BOOL | 0 |

## See also

Description of FlipFlop (Page 2409)
FlipFlop modes (Page 2412)
FlipFlop functions (Page 2412)
FlipFlop error handling (Page 2413)
FlipFlop messaging (Page 2414)
FlipFlop block diagram (Page 2416)

## 15.3.7 FlipFlop block diagram

### FlipFlop block diagram

A block diagram is not provided for this block.

### See also

Description of FlipFlop (Page 2409)
FlipFlop modes (Page 2412)
FlipFlop functions (Page 2412)
FlipFlop error handling (Page 2413)
FlipFlop messaging (Page 2414)
FlipFlop I/Os (Page 2414)

# 15.4 Or04 - Forming an OR signal from 4 binary input signals

## 15.4.1 Description of Or04

### Object name (type + number) and family

Type + number: FC 364

Family: LogicDi

### Area of application for Or04

The block is used for the following applications:

- Forming an OR-output signal from four binary input values

### How it works

Four input parameters are combined via the OR-function ("or-ing") into one output value `Out`.

You can find additional information on forming the signal status under Forming and outputting the signal status of digital logic blocks (Page 119)

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Or04 block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Cascade control with control loop monitoring through ConPerMon (CascadeControl) (Page 2840)
- Cascade control with PIDConR (CascadeR) (Page 2843)
- Dosing (Dose_Lean) (Page 2856)
- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 2827)
- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 2825)
- Model-based predictive control (ModPreCon) (Page 2849)
- Override control (Page 2846)
- Override control with PIDConR (OverrideR) (Page 2848)
- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 2822)
- PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 2823)
- Ratio control with control loop monitoring through ConPerMon (RatioControl) (Page 2836)

- Ratio control with PIDConR (RatioR) (Page 2839)
- PID controller with Smith predictor (SmithPredictorControl) (Page 2830)
- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 2832)
- Valve (Valve_Lean) (Page 2865)
- Dosing with PA/FF devices (Dose_Lean_Fb) (Page 2857)
- PID controller for PA/FF devices (PIDControl_Lean_Fb) (Page 2821)
- Control valve (VlvAnL) (Page 2868)
- Control valve for PA/FF devices (ValveAnalog_Fb) (Page 2869)
- Split-range controller with control loop monitoring through ConPerMon (SplitrangeControl) (Page 2833)

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

### See also

Or04 block diagram (Page 2421)

Or04 I/Os (Page 2420)

Or04 messaging (Page 2420)

Or04 error handling (Page 2419)

Or04 functions (Page 2419)

Or04 modes (Page 2418)

## 15.4.2 Or04 modes

### Or04 modes

This block does not have any modes.

### See also

Or04 block diagram (Page 2421)

Or04 I/Os (Page 2420)

Or04 messaging (Page 2420)
Or04 error handling (Page 2419)
Or04 functions (Page 2419)
Description of Or04 (Page 2417)

## 15.4.3 Or04 functions

### Functions of Or04

This block does not have any other functions.

### See also

Or04 block diagram (Page 2421)
Or04 I/Os (Page 2420)
Or04 messaging (Page 2420)
Or04 error handling (Page 2419)
Or04 modes (Page 2418)
Description of Or04 (Page 2417)

## 15.4.4 Or04 error handling

### Or04 error handling

The block does not report any errors.

### See also

Or04 block diagram (Page 2421)
Or04 I/Os (Page 2420)
Or04 messaging (Page 2420)
Or04 functions (Page 2419)
Or04 modes (Page 2418)
Description of Or04 (Page 2417)

## 15.4.5 Or04 messaging

### Messaging

This block does not offer messaging.

### See also

Or04 block diagram (Page 2421)
Or04 I/Os (Page 2420)
Or04 error handling (Page 2419)
Or04 functions (Page 2419)
Or04 modes (Page 2418)
Description of Or04 (Page 2417)

## 15.4.6 Or04 I/Os

### Or04 I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In1 | Value 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| In2 | Value 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| In3 | Value 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| In4 | Value 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### See also

Or04 block diagram (Page 2421)
Or04 messaging (Page 2420)
Or04 error handling (Page 2419)
Or04 functions (Page 2419)
Or04 modes (Page 2418)
Description of Or04 (Page 2417)

## 15.4.7 Or04 block diagram

### Or04 block diagram

A block diagram is not provided for this block.

### See also

Or04 I/Os (Page 2420)
Or04 messaging (Page 2420)
Or04 error handling (Page 2419)
Or04 functions (Page 2419)
Or04 modes (Page 2418)
Description of Or04 (Page 2417)

# 15.5 Or08 - Forming an OR signal from 8 binary input signals

## 15.5.1 Description of Or08

### Object name (type + number) and family

Type + number: FC 365

Family: LogicDi

### Area of application for Or08

The block is used for the following applications:

- Forming an OR- output signal from eight binary input values

### How it works

Eight input parameters are combined via the OR-function ("or-ing") into one output value `Out`.

You can find additional information on forming the signal status under Forming and outputting the signal status of digital logic blocks (Page 119)

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Or08 block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Monitoring eight digital process tags (Digital8Monitoring) (Page 2853)
- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 2832)
- Step controller with direct access to the actuator and without position feedback (StepControlDirect) (Page 2831)
- Two-speed motor (Motor2Speed) (Page 2860)
- Reversing motor (MotorReversible) (Page 2861)
- Reversible motor with controllable speed (MotorSpeedControlled) (Page 2862)
- Two-way valve (Valve2Way) (Page 2866)
- Motor valve (ValveMotor) (Page 2867)

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

### See also

Or08 block diagram (Page 2426)
Or08 I/Os (Page 2425)
Or08 messaging (Page 2424)
Or08 error handling (Page 2424)
Or08 functions (Page 2423)
Or08 modes (Page 2423)

## 15.5.2 Or08 modes

### Or08 modes

This block does not have any operating modes.

### See also

Or08 block diagram (Page 2426)
Or08 I/Os (Page 2425)
Or08 messaging (Page 2424)
Or08 error handling (Page 2424)
Or08 functions (Page 2423)
Description of Or08 (Page 2422)

## 15.5.3 Or08 functions

### Functions of Or08

This block does not have any other functions.

### See also

Or08 block diagram (Page 2426)
Or08 I/Os (Page 2425)

Or08 messaging (Page 2424)
Or08 error handling (Page 2424)
Or08 modes (Page 2423)
Description of Or08 (Page 2422)

## 15.5.4 Or08 error handling

### Or08 error handling

The block does not report any errors.

### See also

Or08 block diagram (Page 2426)
Or08 I/Os (Page 2425)
Or08 messaging (Page 2424)
Or08 functions (Page 2423)
Or08 modes (Page 2423)
Description of Or08 (Page 2422)

## 15.5.5 Or08 messaging

### Messaging

This block does not offer messaging.

### See also

Or08 block diagram (Page 2426)
Or08 I/Os (Page 2425)
Or08 error handling (Page 2424)
Or08 functions (Page 2423)
Or08 modes (Page 2423)
Description of Or08 (Page 2422)

## 15.5.6 Or08 I/Os

### Or08 I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In1 | Value 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| In2 | Value 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| In3 | Value 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| In4 | Value 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| In5 | Value 5 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| In6 | Value 6 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| In7 | Value 7 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| In8 | Value 8 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### See also

Or08 block diagram (Page 2426)
Or08 messaging (Page 2424)
Or08 error handling (Page 2424)
Or08 functions (Page 2423)
Or08 modes (Page 2423)
Description of Or08 (Page 2422)

## 15.5.7 Or08 block diagram

### Or08 block diagram

A block diagram is not provided for this block.

### See also

Or08 I/Os (Page 2425)
Or08 messaging (Page 2424)
Or08 error handling (Page 2424)
Or08 functions (Page 2423)
Or08 modes (Page 2423)
Description of Or08 (Page 2422)

# 15.6 Not01 - Inversion of an input signal

## 15.6.1 Description of Not01

### Object name (type + number) and family

Type + number: FC 382

Family: LogicDi

### Area of application for Not01

The block is used for the following applications:

- Inversion of an input signal

### How it works

The block inverts the binary signal available at the `In` input parameter and writes the result to its output parameter `Out`.

The signal status is passed from the input directly to the output.

You can find additional information on forming the signal status under Forming and outputting the signal status of digital logic blocks (Page 119)

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Not01 block, the Advanced Process Library contains a template for process tag types as an example with an application scenario for this block.

Example of process tag types:

- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 2832)
- Ratio control with control loop monitoring through ConPerMon (RatioControl) (Page 2836)

### See also

Not01 block diagram (Page 2430)

Not01 I/Os (Page 2429)

Not01 messaging (Page 2429)

Not01 error handling (Page 2428)

Not01 functions (Page 2428)

Not01 modes (Page 2428)

## 15.6.2 Not01 modes

### Not01 operating modes

This block does not have any operating modes.

### See also

Not01 block diagram (Page 2430)
Not01 I/Os (Page 2429)
Not01 messaging (Page 2429)
Not01 error handling (Page 2428)
Not01 functions (Page 2428)
Description of Not01 (Page 2427)

## 15.6.3 Not01 functions

### Functions of Not01

There are no other functions for this block.

### See also

Not01 block diagram (Page 2430)
Not01 I/Os (Page 2429)
Not01 messaging (Page 2429)
Not01 error handling (Page 2428)
Not01 modes (Page 2428)
Description of Not01 (Page 2427)

## 15.6.4 Not01 error handling

### Not01 error handling

The block does not report any errors.

### See also

Not01 block diagram (Page 2430)
Not01 I/Os (Page 2429)
Not01 messaging (Page 2429)
Not01 functions (Page 2428)
Not01 modes (Page 2428)
Description of Not01 (Page 2427)

## 15.6.5 Not01 messaging

### Messaging

This block does not offer messaging.

### See also

Not01 block diagram (Page 2430)
Not01 I/Os (Page 2429)
Not01 error handling (Page 2428)
Not01 functions (Page 2428)
Not01 modes (Page 2428)
Description of Not01 (Page 2427)

## 15.6.6 Not01 I/Os

### Not01 I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In | Input value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### See also

Not01 block diagram (Page 2430)
Not01 messaging (Page 2429)
Not01 error handling (Page 2428)
Not01 functions (Page 2428)
Not01 modes (Page 2428)
Description of Not01 (Page 2427)

## 15.6.7 Not01 block diagram

### Not01 block diagram

A block diagram is not provided for this block.

### See also

Not01 I/Os (Page 2429)
Not01 messaging (Page 2429)
Not01 error handling (Page 2428)
Not01 functions (Page 2428)
Not01 modes (Page 2428)
Description of Not01 (Page 2427)

# 15.7 RedDi02 - 1 out of 2 selection for redundant digital values

## 15.7.1 Description of RedDi02

### Object name (type + number) and family

Type + number: FC 386

Family: LogicDi

### Area of application for RedDi02

The block is used for the following applications:

- 1 out of 2 selection for redundant digital values

### How it works

The block selects from two input values the one with the best signal status and outputs it at the output `Out`. In addition, the outputs `SimAct`, `Uncertain` and `LossRed` are set according to the signal status.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

### See also

RedDi02 modes (Page 2432)

RedDi02 functions (Page 2432)

RedDi02 error handling (Page 2433)

RedDi02 messaging (Page 2433)

RedDi02 I/Os (Page 2433)

RedDi02 block diagram (Page 2435)

## 15.7.2 RedDi02 modes

### RedDi02 operating modes

This block does not have any operating modes.

### See also

Description of RedDi02 (Page 2431)
RedDi02 functions (Page 2432)
RedDi02 error handling (Page 2433)
RedDi02 messaging (Page 2433)
RedDi02 I/Os (Page 2433)
RedDi02 block diagram (Page 2435)

## 15.7.3 RedDi02 functions

### Functions of RedDi02

The functions for this block are listed below.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status of redundancy blocks (Page 121).

### See also

Description of RedDi02 (Page 2431)
RedDi02 modes (Page 2432)
RedDi02 error handling (Page 2433)
RedDi02 messaging (Page 2433)
RedDi02 I/Os (Page 2433)
RedDi02 block diagram (Page 2435)

## 15.7.4 RedDi02 error handling

### RedDi02 error handling

The block does not report any errors.

### See also

Description of RedDi02 (Page 2431)
RedDi02 modes (Page 2432)
RedDi02 functions (Page 2432)
RedDi02 messaging (Page 2433)
RedDi02 I/Os (Page 2433)
RedDi02 block diagram (Page 2435)

## 15.7.5 RedDi02 messaging

### Messaging

This block does not offer messaging.

### See also

Description of RedDi02 (Page 2431)
RedDi02 modes (Page 2432)
RedDi02 functions (Page 2432)
RedDi02 error handling (Page 2433)
RedDi02 I/Os (Page 2433)
RedDi02 block diagram (Page 2435)

## 15.7.6 RedDi02 I/Os

### I/Os of RedDi02

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In1 | Digital input value 1 | STRUCT<br>• Value: BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| In2 | Digital input value 2 | STRUCT<br>• Value: BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| LossRed | 1= Redundancy loss at one of the inputs | STRUCT<br>• Value: BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Out | Output of the process value with the better signal status | STRUCT<br>• Value: BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1 = one input value has the Simulation status | STRUCT<br>• Value: BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |
| Uncertain | 1 = one input value has the "uncertain" status | STRUCT<br>• Value: BOOL<br>• ST:BYTE | -<br>• 0<br>• 16#80 |

### See also

Description of RedDi02 (Page 2431)
RedDi02 modes (Page 2432)
RedDi02 functions (Page 2432)
RedDi02 error handling (Page 2433)
RedDi02 messaging (Page 2433)
RedDi02 block diagram (Page 2435)

## 15.7.7 RedDi02 block diagram

### RedDi02 block diagram

A block diagram is not provided for this block.

### See also

Description of RedDi02 (Page 2431)
RedDi02 modes (Page 2432)
RedDi02 functions (Page 2432)
RedDi02 error handling (Page 2433)
RedDi02 messaging (Page 2433)
RedDi02 I/Os (Page 2433)

# 15.8 SelD02In - Output of one of two digital signals

## 15.8.1 Description of SelD02In

### Object name (type + number) and family

Type + number: FC 391

Family: LogicDi

### Area of application for SelD02In

The block is used for the following applications:

- Selection from two digital values

### How it works

Depending on the value of the input `Sel_In2`, the block switches the value of the input `In1` or the input `In2` to the output `Out`.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

### See also

SelD02In modes (Page 2437)

SelD02In functions (Page 2437)

SelD02In error handling (Page 2438)

SelD02In messaging (Page 2438)

SelD02In I/Os (Page 2439)

SelD02In block diagram (Page 2440)

## 15.8.2 SelD02In modes

### SelD02In operating modes

This block does not have any operating modes.

### See also

Description of SelD02In (Page 2436)
SelD02In functions (Page 2437)
SelD02In error handling (Page 2438)
SelD02In messaging (Page 2438)
SelD02In I/Os (Page 2439)
SelD02In block diagram (Page 2440)

## 15.8.3 SelD02In functions

### Functions of SelD02In

The functions for this block are listed below.

### Select input parameter

You can use the parameter `Sel_In2` to specify whether the input parameter `In1` or `In2` is to be output at the output parameter:

- `Sel_In2` = 0: Input parameter `In1` is written with its signal status to output parameter `Out`.
- `Sel_In2` = 1: Input parameter `In2` is written with its signal status to output parameter `Out`.

### Display of selected value

The output parameter `In2Selected` indicates which of the two input parameters has just been output:

- `In2Selected` = 0: At the output parameter `Out`, the value of the input parameter `In1` is output.
- `In2Selected` = 1: At the output parameter `Out`, the value of the input parameter `In2` is output.

The signal status of `Sel_In2` is output at the `In2Selected` output parameter.

**See also**

Description of SelD02In (Page 2436)
SelD02In modes (Page 2437)
SelD02In error handling (Page 2438)
SelD02In messaging (Page 2438)
SelD02In I/Os (Page 2439)
SelD02In block diagram (Page 2440)

## 15.8.4 SelD02In error handling

**SelD02In error handling**

The block does not report any errors.

**See also**

Description of SelD02In (Page 2436)
SelD02In modes (Page 2437)
SelD02In functions (Page 2437)
SelD02In messaging (Page 2438)
SelD02In I/Os (Page 2439)
SelD02In block diagram (Page 2440)

## 15.8.5 SelD02In messaging

**Messaging**

This block does not offer messaging.

**See also**

Description of SelD02In (Page 2436)
SelD02In modes (Page 2437)
SelD02In functions (Page 2437)
SelD02In error handling (Page 2438)
SelD02In I/Os (Page 2439)
SelD02In block diagram (Page 2440)

## 15.8.6 SelD02In I/Os

### SelD02In I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In1 | Input 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| In2 | Input 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Sel_In2 | Selecting the input parameter:<br>0 = In1<br>1 = In2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| In2Selected | Selected input parameter:<br>0 = In1<br>1 = In2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Out | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### See also

Description of SelD02In (Page 2436)
SelD02In modes (Page 2437)
SelD02In functions (Page 2437)
SelD02In error handling (Page 2438)
SelD02In messaging (Page 2438)
SelD02In block diagram (Page 2440)

## 15.8.7 SelD02In block diagram

### SelD02In block diagram

A block diagram is not provided for this block.

### See also

Description of SelD02In (Page 2436)
SelD02In modes (Page 2437)
SelD02In functions (Page 2437)
SelD02In error handling (Page 2438)
SelD02In messaging (Page 2438)
SelD02In I/Os (Page 2439)

# 15.9 StrctCom - Structure composer for 32-bit structures

## 15.9.1 Description of StrctCom

### Object name (type + number) and family

Type + number: FC 393

Family: LogicDi

### Area of application for StrctCom

The block forms a structure type output with 32 binary structural elements from 32 binary input signals. The structure corresponds, for example, to the `OS_Perm` inputs of the technological blocks and can be interconnected to them.

### How it works

The block is used for create a structure of 32 input bits.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

### See also

StrctCom block diagram (Page 2444)

StrctCom I/Os (Page 2443)

StrctCom messaging (Page 2443)

StrctCom error handling (Page 2443)

StrctCom functions (Page 2442)

StrctCom modes (Page 2442)

Forming and outputting the signal status of digital logic blocks (Page 119)

## 15.9.2 StrctCom modes

### StrctCom operating modes

This block does not have any operating modes.

### See also

StrctCom block diagram (Page 2444)
StrctCom I/Os (Page 2443)
StrctCom messaging (Page 2443)
StrctCom error handling (Page 2443)
StrctCom functions (Page 2442)
Description of StrctCom (Page 2441)

## 15.9.3 StrctCom functions

### Functions of StrctCom

The functions for this block are listed below.

Any 32 bits at the input are assigned to the structure in the output.

```
Out.Bit0 = Bit0
Out.Bit1 = Bit1
  .....
Out.Bit31 = Bit31
```

### See also

StrctCom block diagram (Page 2444)
StrctCom I/Os (Page 2443)
StrctCom messaging (Page 2443)
StrctCom error handling (Page 2443)
StrctCom modes (Page 2442)
Description of StrctCom (Page 2441)

## 15.9.4 StrctCom error handling

### Error handling of StrctCom

The block does not report any errors.

### See also

StrctCom block diagram (Page 2444)
StrctCom I/Os (Page 2443)
StrctCom messaging (Page 2443)
StrctCom functions (Page 2442)
StrctCom modes (Page 2442)
Description of StrctCom (Page 2441)

## 15.9.5 StrctCom messaging

### Messaging

This block does not offer messaging.

### See also

StrctCom block diagram (Page 2444)
StrctCom I/Os (Page 2443)
StrctCom error handling (Page 2443)
StrctCom functions (Page 2442)
StrctCom modes (Page 2442)
Description of StrctCom (Page 2441)

## 15.9.6 StrctCom I/Os

I/Os of StrctCom

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `Bit0...Bit31` | Binary input value Bit0...Bit 31 | BOOL | 0 |
| `EN` | 1 = Called block will be processed | BOOL | 1 |
| `IN` | Binary input value<br>• Bit 0 .. 31 | <br>• BOOL | |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `ENO` | 1 = Block algorithm completed without errors | BOOL | 0 |
| `Out` | Structure of 0...31 binary signals | STRUCT<br>• Bit0: BOOL<br>• ...<br>• Bit31: BOOL | -<br>• 0 |

### See also

StrctCom block diagram (Page 2444)
StrctCom messaging (Page 2443)
StrctCom error handling (Page 2443)
StrctCom functions (Page 2442)
StrctCom modes (Page 2442)
Description of StrctCom (Page 2441)

## 15.9.7 StrctCom block diagram

### Block diagram of StrctCom

A block diagram is not provided for this block.

### See also

StrctCom I/Os (Page 2443)
StrctCom messaging (Page 2443)
StrctCom error handling (Page 2443)
StrctCom functions (Page 2442)

StrctCom modes (Page 2442)
Description of StrctCom (Page 2441)

# 15.10 StrctDeC - Structure decomposer for 32-bit structures

## 15.10.1 Description of StrctDeC

### Object name (type + number) and family

Type + number: FC 396

Family: LogicDi

### Area of application for StrctDeC

The block forms a DWORD type output and 32 single binary outputs from a structure type input with 32 binary structural elements. The input structure corresponds, for example, to the `OS_Perm` inputs of the technological blocks. If you use an `OS_Perm` input in a CFC source chart, you can use this block to transform the structured input in separate binary values inside the CFC source chart. For more information, refer to the Source chart for GainSched function block (gain scheduling) (Page 2845).

### How it works

The block is used to create a DWORD or single binary outputs from the structure of 32 input bits.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

StrctDeC modes (Page 2447)

StrctDeC functions (Page 2447)

StrctDeC error handling (Page 2448)

StrctDeC messaging (Page 2448)

StrctDeC I/Os (Page 2448)

StrctDeC block diagram (Page 2449)

## 15.10.2 StrctDeC modes

### StrctDeC operating modes

This block does not have any operating modes.

### See also

Description of StrctDeC (Page 2446)
StrctDeC functions (Page 2447)
StrctDeC error handling (Page 2448)
StrctDeC messaging (Page 2448)
StrctDeC I/Os (Page 2448)
StrctDeC block diagram (Page 2449)

## 15.10.3 StrctDeC functions

### Functions of StrctDeC

The functions for this block are listed below.

The input structure `In` is assigned to a DWORD output and to 32 single binary outputs.

```
OutDW = Structure In
Bit0 = In.Bit0
Bit1 = In.Bit1
...
Bit31 = In.Bit31
```

### See also

Description of StrctDeC (Page 2446)
StrctDeC modes (Page 2447)
StrctDeC error handling (Page 2448)
StrctDeC messaging (Page 2448)
StrctDeC I/Os (Page 2448)
StrctDeC block diagram (Page 2449)

## 15.10.4 StrctDeC error handling

### Error handling of StrctDeC

The block does not report any errors.

### See also

Description of StrctDeC (Page 2446)
StrctDeC modes (Page 2447)
StrctDeC functions (Page 2447)
StrctDeC messaging (Page 2448)
StrctDeC I/Os (Page 2448)
StrctDeC block diagram (Page 2449)

## 15.10.5 StrctDeC messaging

### Messaging

This block does not offer messaging.

### See also

Description of StrctDeC (Page 2446)
StrctDeC modes (Page 2447)
StrctDeC functions (Page 2447)
StrctDeC error handling (Page 2448)
StrctDeC I/Os (Page 2448)
StrctDeC block diagram (Page 2449)

## 15.10.6 StrctDeC I/Os

### I/Os of StrctDeC

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In | Structure of 0...31 binary signals Bit 0...31 | STRUCT<br>• Bit0: BOOL<br>• ...<br>• Bit31: BOOL | -<br>• 0<br>• 0<br>• 0 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| OutDW | DWORD output value | DWORD | 16#00000000 |
| Bit0 | Binary output value | Bit0: BOOL | 0 |
| ... | ... | ... | ... |
| Bit31 | Binary output value | Bit0: BOOL | 0 |

### See also

Description of StrctDeC (Page 2446)
StrctDeC modes (Page 2447)
StrctDeC functions (Page 2447)
StrctDeC error handling (Page 2448)
StrctDeC messaging (Page 2448)
StrctDeC block diagram (Page 2449)

## 15.10.7 StrctDeC block diagram

### Block diagram of StrctDeC

A block diagram is not provided for this block.

### See also

Description of StrctDeC (Page 2446)
StrctDeC modes (Page 2447)
StrctDeC functions (Page 2447)
StrctDeC error handling (Page 2448)

StrctDeC messaging (Page 2448)
StrctDeC I/Os (Page 2448)

# 15.11 Trigger - Detection of rising and falling edges

## 15.11.1 Description of Trigger

### Object name (type + number) and family

Type + number: FB 1821

Family: LogicDi

### Area of application for Trigger

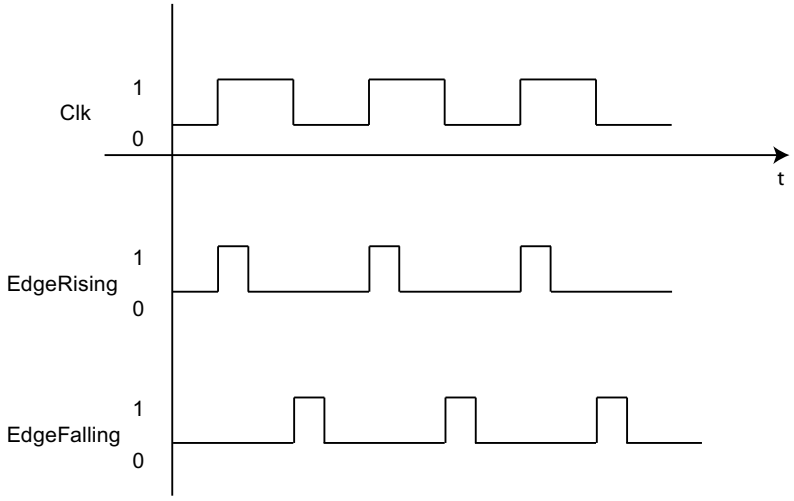
The block is used for the following applications:

- Detection of rising and falling edges

### How it works

This block checks the input variable Clk for occurrences of rising and falling edges, and signals them at the outputs `EdgeRising` and `EdgeFalling` .

Output `EdgeRising` is set to 1 for a block cycle at a rising edge. Output `EdgeFalling` is set to 1 for a block cycle at a falling edge.




### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

Edge detection is suppressed at startup or initial startup.

With the input parameter `InitOnRestart`, you can influence the startup behavior of the output parameters `InitOut`, `EdgeRising`, and `EdgeFalling`:

- `InitOnRestart` = 0 (Default):
  The output parameters `EdgeRising` and `EdgeFalling` remain unchanged and the output parameter `InitOut` will reset to false.
- `InitOnRestart` = 1:
  The output parameters `EdgeRising` and `EdgeFalling` will reset to false and the output parameter `InitOut` will set to true. In the second cycle of OB30 to OB38, the output parameter `InitOut` will be reset. You can connect this output parameter with the input parameter `Init` of the block Limit, TimerP, or FlipFlop to initialize these blocks. For more information, refer to Startup characteristics over Trigger block (Page 75).

After a warm start of CPU with `Clk =1` and `InitOnRestart =1`, the output EdgeRising will set for one cycle.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

Trigger modes (Page 2452)
Trigger functions (Page 2453)
Trigger error handling (Page 2453)
Trigger messaging (Page 2454)
Trigger I/Os (Page 2454)
Trigger block diagram (Page 2455)

## 15.11.2 Trigger modes

### Trigger operating modes

This block does not have any operating modes.

### See also

Description of Trigger (Page 2451)
Trigger functions (Page 2453)
Trigger error handling (Page 2453)
Trigger messaging (Page 2454)
Trigger I/Os (Page 2454)
Trigger block diagram (Page 2455)

## 15.11.3 Trigger functions

### Functions of Trigger

This block provides the function:

### Forming the signal status for blocks

The `Clk.ST` status is transmitted with each cycle to `EdgeRising.ST` and `EdgeFalling.ST`.

Trigger forms the signal status as follows:

- `EdgeRising.ST` := `Clk.ST`
- `EdgeFalling.ST` := `Clk.ST`

### See also

Description of Trigger (Page 2451)
Trigger modes (Page 2452)
Trigger error handling (Page 2453)
Trigger messaging (Page 2454)
Trigger I/Os (Page 2454)
Trigger block diagram (Page 2455)

## 15.11.4 Trigger error handling

### Error handling of Trigger

The block does not report any errors.

### See also

Description of Trigger (Page 2451)
Trigger modes (Page 2452)
Trigger functions (Page 2453)
Trigger messaging (Page 2454)
Trigger I/Os (Page 2454)
Trigger block diagram (Page 2455)

## 15.11.5 Trigger messaging

### Messaging

This block does not offer messaging.

### See also

Description of Trigger (Page 2451)
Trigger modes (Page 2452)
Trigger functions (Page 2453)
Trigger error handling (Page 2453)
Trigger I/Os (Page 2454)
Trigger block diagram (Page 2455)

## 15.11.6 Trigger I/Os

### SelD02In I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Clk | Input | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| InitOnRestart | 1 = Initialization on warm restart | BOOL | 0 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| EdgeRising | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| EdgeFalling | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| InitOut | Recognition of warm restart (OB100) | BOOL | 0 |

### See also

Description of Trigger (Page 2451)
Trigger modes (Page 2452)
Trigger functions (Page 2453)
Trigger error handling (Page 2453)
Trigger messaging (Page 2454)
Trigger block diagram (Page 2455)

## 15.11.7 Trigger block diagram

### Trigger block diagram

A block diagram is not provided for this block.

### See also

Description of Trigger (Page 2451)
Trigger modes (Page 2452)
Trigger functions (Page 2453)
Trigger error handling (Page 2453)
Trigger messaging (Page 2454)
Trigger I/Os (Page 2454)

# 15.12 XOr04 - EXCLUSIVE OR logic operation

## 15.12.1 Description of XOr04

### Object name (type + number) and family

Type + number: FC 388

Family: LogicDi

### Area of application for XOr04

The block is used for the following applications:

- EXCLUSIVE OR logic operation with up to 4 inputs

### How it works

The block performs a logic operation with up to four input parameters `In1` to `In4`. The output parameter `Out` then becomes exactly 1, if there is a 1 at an uneven number of inputs and 0 at the rest of them.

### Truth table

| | In1 | In2 | In3 | In4 | Out |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 0 |
| 6 | 0 | 1 | 1 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 1 |
| 9 | 1 | 0 | 0 | 1 | 0 |
| 10 | 1 | 0 | 1 | 0 | 0 |
| 11 | 1 | 0 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 | 1 |
| 14 | 1 | 1 | 1 | 0 | 1 |
| 15 | 1 | 1 | 1 | 1 | 0 |

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

XOr04 modes (Page 2457)
XOr04 functions (Page 2457)
XOr04 error handling (Page 2458)
XOr04 messaging (Page 2458)
XOr04 I/Os (Page 2459)
XOr04 block diagram (Page 2460)

## 15.12.2 XOr04 modes

### XOr04 operating modes

This block does not have any operating modes.

### See also

Description of XOr04 (Page 2456)
XOr04 functions (Page 2457)
XOr04 error handling (Page 2458)
XOr04 messaging (Page 2458)
XOr04 I/Os (Page 2459)
XOr04 block diagram (Page 2460)

## 15.12.3 XOr04 functions

### Functions of XOr04

The functions for this block are listed below.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status of digital logic blocks (Page 119).

### See also

Description of XOr04 (Page 2456)
XOr04 modes (Page 2457)
XOr04 error handling (Page 2458)
XOr04 messaging (Page 2458)
XOr04 I/Os (Page 2459)
XOr04 block diagram (Page 2460)

## 15.12.4 XOr04 error handling

### XOr04 error handling

The block does not report any errors.

### See also

Description of XOr04 (Page 2456)
XOr04 modes (Page 2457)
XOr04 functions (Page 2457)
XOr04 messaging (Page 2458)
XOr04 I/Os (Page 2459)
XOr04 block diagram (Page 2460)

## 15.12.5 XOr04 messaging

### Messaging

This block does not offer messaging.

### See also

Description of XOr04 (Page 2456)
XOr04 modes (Page 2457)
XOr04 functions (Page 2457)

XOr04 error handling (Page 2458)
XOr04 I/Os (Page 2459)
XOr04 block diagram (Page 2460)

## 15.12.6 XOr04 I/Os

### XOr04 I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In1 | Input 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| In2 | Input 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| In3 | Input 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| In4 | Input 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Output | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### See also

Description of XOr04 (Page 2456)
XOr04 modes (Page 2457)
XOr04 functions (Page 2457)

XOr04 error handling (Page 2458)
XOr04 messaging (Page 2458)
XOr04 block diagram (Page 2460)

## 15.12.7 XOr04 block diagram

### XOr04 block diagram

A block diagram is not provided for this block.

### See also

Description of XOr04 (Page 2456)
XOr04 modes (Page 2457)
XOr04 functions (Page 2457)
XOr04 error handling (Page 2458)
XOr04 messaging (Page 2458)
XOr04 I/Os (Page 2459)

# Generator blocks 16

## 16.1 NoiseGen - Generating signal noise

### 16.1.1 Description of NoiseGen

#### Object name (type + number) and family

Object name: FB 1863

Family: Genrator

#### Area of application for NoiseGen

The block is used for the following applications:

- Noise generator

The block can only be used for the simulation in the example project.

#### How it works

This block serves to generate signal noise. It is used for demonstration purposes in the example project so that the simulated signals react naturally.

You can employ it for simulation examples, demonstrations, trade fair models etc. It is never needed in real plants because real measuring devices always supply signals with noise.

There is an example project for the NoiseGen block (APL_Example_xx, xx refers to the language variant) with an application scenario for this block, which explains how the block works.

Application scenario in the example project:

- Process simulation including noise generator (ProcSimC; ProcSimS) (Page 2871)

#### See also

NoiseGen I/Os (Page 2461)

### 16.1.2 NoiseGen I/Os

#### NoiseGen I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Enable | 1 = generate noise signals | BOOL | 1 |
| Offset | Mean temporal value of the Noise output signal | REAL | 20.0 |
| Restart* | 1 = block restart | BOOL | 1 |
| StdDev | Standard deviation of the noise signal | REAL | 1.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Noise | Generated noise signal | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### See also

Description of NoiseGen (Page 2461)

# Channel blocks 17

## 17.1 Information on using channel blocks

### Information on using channel blocks

- The descriptions of the channel blocks specify the OBs in which the blocks are installed. Please note that not all OBs listed will be generated for all CPUs. You can find additional information in the online help of the particular OB.
- If the driver generator uses the channel blocks of the PCS 7 libraries, you require firmware version V3.1 or higher on the CPU.
- The CFC function "Generate module drivers" interconnects and configures the required I/Os automatically. The function is called and executed if hardware modifications are detected when compiling the program, for example.
- The channel blocks (except Pcs7HaAI and Pcs7HaAO) evaluate the new first run identifier of the diagnostic function blocks from the PCS 7 Basis Library connected to the `DataXchg1` input.

| **NOTICE** |
| --- |
| **Behavior change in the following cases:**<br>• **when not using the CFC function "Generate module driver"**<br>• **when using 3rd-Party diagnostic blocks**<br>• **when using the function block without `DataXchg1` connection to Basis Library function blocks**<br>Usually, the `DataXchg1` input of the channel blocks is automatically connected when the module drivers are generated. Users who do not use this function must configure the `Feature.Bit27` Do not evaluate first run identifier of the diagnostic blocks (Page 205) with "1" at the respective channel blocks.<br>The `Feature.Bit27` must also be set if 3rd-Party diagnostic blocks are used that do not provide a first run identifier or in the case where the channel block is used without `DataXchg1` connection. |

- The channel blocks support the HART communication protocol. To use the HART protocol, you must ensure that it is activated in the Hardware Configuration of the module.

## Signal-processing blocks

Various types of channel blocks are provided in this PCS 7 library for processing signals from inputs and outputs:

1. **Standard channel blocks**
   This applies to the following blocks:
   - Pcs7AnIn
   - Pcs7AnOu
   - Pcs7Cnt1
   - Pcs7Cnt2
   - Pcs7Cnt3
   - Pcs7DiIn
   - Pcs7DiIT
   - Pcs7DiOu
   - Pcs7HaAI
   - Pcs7HaAO
   - Pcs7AITC
   - Pcs7DiCnt
   - Pcs7Cnt4
   - Pcs7ReIn

   These blocks are used only for processing the signals of S7-300/400 SM modules. Use these standard blocks if you want to optimize memory and runtime utilization and do not need to process any PA devices.

2. **FF/PA channel blocks**
   This applies to the following blocks:
   - FbAnIn
   - FbAnOu
   - FbAnTot
   - FbDiIn
   - FbDiOu
   - FbDrive
   - FbEnMe
   - FbSwtMMS

   These blocks are designed especially for use with PA field devices and the PROFIBUS 3.0 Class A and B or with FF field devices. In particular, you should use these blocks if you want to make use of the special features of these devices. In contrast to standard channel blocks, PA channel blocks not only process the signal itself but also all variables, according to the desired device configuration selected in the hardware configuration.

## MS_Ext (External maintenance status)

The MS_Ext is used to combine an external maintenance like an operation hour counter with the maintenance state of the device. Activate Feature.Bit10 on the channel block and interconnect the MS_Ext input on the channel block with the signals of the external maintenance state. The MSTOu converter block can be used for this.

**Note**

The external maintenance statuses "Maintenance demanded" (corresponding to significance 6) and "Maintenance required" (corresponding to significance 5) only are taken into account in the diagnostics block. These maintenance statuses trigger the messages "Maintenance demanded" (external) and "Maintenance required"(external) in the process control.

MS_Ext supports the following states:

- Maintenance alarm: 16#07
- Function control (Simulation): 16#03

and MS = 7 and MS = 3 must be unblocked in all function blocks

**Text_Ref**
The operator has the possibility to set few external diagnostics information in the maintenance object of the module/device. Therefore, interconnect the Text_Ref input on the channel block with the signals used for displaying the external diagnostic information. The BO_W converter block can be used for this. The individual bits of the Text_Ref input activates the state display in the faceplate and the "External diagnostics" view in process mode. The text must be defined by the operator in the "Advanced Diagnostic" settings.
For more details, refer the *PCS 7 Maintenance Station manual*.

## Channel active information

Channel active information (byte 0, bit 3 in DataXchg1):

0 = channel is deactivated in HW Config.

1 = channel is activated in HW Config.

In case the channel is deactivated, the channel blocks deliver signal state 16#68 (uncertain, maintenance demand) and Bad=false.

# 17.2 FbAnIn - Analog input channel block for field devices

## 17.2.1 Description of FbAnIn

### Object name (type + number) and family

Type + number: FB 1813

Family: Channel

### Area of application for FbAnIn

The block is used for the following applications:

- Signal processing (cyclic service) in accordance with "Transmitter" PROFIBUS PA profile of an analog input value:
  - Of a PA field device in accordance with PROFIBUS 3.0 class A and B
  - Of an auxiliary variable of a HART field device
  - Of an FF field device
- Processing of the redundant input channels.

### How it works

Block FbAnIn cyclically reads the process value and the signal status of the field device from the process image (partition). The process value is available as a physical variable. The signal status contains information about the status of the field device.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Connect the symbol generated in HW Config (symbol table) for the input channel with the `PV` input parameter.

In case of redundant I/O configuration, use symbol of the channel of the master module.

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameter `Mode` is interconnected with the corresponding output parameter `OMODE_xx` of the MOD_PAL0, FF_MOD32- or MOD_PAX0 block.
- The in/out parameter `DataXchg` is interconnected with the corresponding output parameter `DXCHG_xx` of the MOD_PAL0, FF_MOD32 - or MOD_PAX0 block.

- The symbol for the signal status of the analog input channel is interconnected to input PV_ST.
- The MS parameter is interconnected to the O_MS output parameter of the diagnostics block.

**Note**

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter Mode manually. Refer to the Mode settings for field devices (Page 2698) chapter for more on this.

In case of redundancy, connection with both quality bits and connection with the redundant slave channel:

- The in parameter PVSlv is interconnected with the symbol of the slave channel.
- The in parameter PV_ST is interconnected with the symbol of the quality byte of the master channel.
- The in parameter PVSlv_ST is interconnected with the symbol of the quality byte of the slave channel.

For the FbAnIn block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Monitoring of an analog process tag for PA/FF devices (AnalogMonitoring_Fb) (Page 2855)
- Dosing with PA/FF devices (Dose_Lean_Fb) (Page 2857)
- PID controller for PA/FF devices (PIDControl_Lean_Fb) (Page 2821)

## Redundancy

The block monitors the values of the redundant signals (master and slave) with quality code and:

- the value of the signal with “Good” quality code along with its quality code is transferred to the process when one of the signals delivers “Good” quality code.
- the value of the master signal along with its quality code is transferred to the process when both the signals deliver “Bad” quality code.

## Startup characteristics

The block does not have any startup characteristics.

## Status word allocation for Status parameter

This block does not have the Status parameter.

**See also**

FbAnIn block diagram (Page 2475)
FbAnIn I/Os (Page 2472)
FbAnIn messaging (Page 2472)
FbAnIn error handling (Page 2471)
FbAnIn functions (Page 2468)
FbAnIn modes (Page 2468)

## 17.2.2 FbAnIn modes

### FbAnIn modes

This block does not have any operating modes.

**See also**

FbAnIn block diagram (Page 2475)
FbAnIn I/Os (Page 2472)
FbAnIn messaging (Page 2472)
FbAnIn error handling (Page 2471)
FbAnIn functions (Page 2468)
Description of FbAnIn (Page 2466)

## 17.2.3 FbAnIn functions

### Functions of FbAnIn

The functions for this block are listed below.

### Obtaining the standard value

The analog value of the process image (partition) is output as a standard value at the `PV_Li` output parameter.

### Holding the last value if raw value is invalid

If the block is to hold its last valid value when the analog value is invalid or in the initialization phase of the device, you must activate this function at the `Feature` Bit Outputting last valid value if raw value is invalid (Page 165).

### Output substitute value if raw value is invalid

If the block is to output a substitute value (SubsPV) when the analog value is invalid or in the initialization phase of the device, you must activate this function at the Feature Bit Output substitute value if raw value is invalid (Page 160).

### Outputting an invalid value if analog value is invalid

If the block is to output an invalid value (PV_Li = PV), you must activate this function at the Feature Bit Output invalid raw value (Page 185).

This function is preset.

### Flutter suppression

This block provides the standard function Flutter suppression for channel blocks (Page 73)

### Signal status for Fb channel blocks

The block provides the standard function Forming and outputting the signal status for channel blocks for field devices (Page 127).

The signal status of process value PV_Li is generated from internal events such as channel errors, higher-level errors, or simulations, as well as from the signal status PV_ST, which comes directly from the device.

The signal status PV_ST can accept values of 16#00 - 16#FF.

The block recognizes a higher-level error, for example, failure of a DP/PA link, via the Mode input parameter.

- If the high byte is Mode = 16#80, then the values in the process image (partition) are valid.
- If the high byte Mode = 16#40 (value status = higher-level error, ModErr = 1), then the analog value is treated as invalid.

The measuring type set in the low word of the Mode input parameter will be ignored.

The bit combinations of signal status PV_ST are output as output parameters (BOOL values). These conform to the bit combinations specified in PROFIBUS 3.0 "General Requirements".

**For FF field devices only:** The values of the signal status PV_ST = 16#84 - 16#87 and 16#90 – 16#93 are evaluated by the link in the same way as signal status 16#80 – 16#83.

If the signal status PV_ST = 16#80 and a process value PV with the value 16#7FFFFFFF (invalid) are transferred from the FF field device, the block will deal with the signal status 16#00 (invalid) from the FF field device.

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 5 | Activate LowCutOff (Page 166) |
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |
| 28 | Output invalid raw value (Page 185) |
| 29 | Output substitute value if raw value is invalid (Page 160) |
| 30 | Outputting last valid value if raw value is invalid (Page 165) |

If `Feature.Bit5` = 1, the input parameter `LowCutOff` is compared with `PV_Out.Value`.

If `PV_Out.Value` < `LowCutOff`, then `PV_Out.Value` = `Scale.Low` and output parameter `PV_LcoAct.Value` = 1.

If `PV_Out.Value` ≥ `LowCutOff`, then `PV_LcoAct.Value` = 0.

### Sign-of-life monitoring

If an input value whose signal status is `16#80` (good) remains constant for a programmable time (monitoring time), the input value is detected as faulty and the outputs `Bad = 1` and `FrzVal = 1` are set.

The monitoring time is set at the `FrznTmIn` input parameter in seconds. With `FrznTmIn = 0` or `FrznEn = 0` (default setting), the sign-of-life monitoring is deactivated, any pending errors are reset.

The input value is considered as faulty as long as it seen as constant. The monitoring time is restarted each time the input value is changed.

### Delay the output "Bad"

If input `DelTiBad` > 0, on receiving an internal bad value, delay timer will start. Timer will abort if `DelTiBad` is 0. Minimum value of `DelTiBad` is 0. Only when timer expires i.e., `RemDelTiBad` = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running `Bad.ST` will be 16#60. Only after the timer expires the actual `ST` be updated. Substitute value is updated at `PV_Out`, only after delay timer expires. When timer is running last valid value will be there at `PV_Out`. When substitute value is used, `ST` of `PV_Out` will be 16#60.

### See also

FbAnIn block diagram (Page 2475)

FbAnIn I/Os (Page 2472)

FbAnIn messaging (Page 2472)

FbAnIn error handling (Page 2471)

FbAnIn modes (Page 2468)
Description of FbAnIn (Page 2466)

## 17.2.4 FbAnIn error handling

### Error handling of FbAnIn

Please refer to the section Error handling (Page 128) in the basic instructions.

The following errors can be displayed for this block:

- Channel error
- Higher-level error
- Invalid measuring range
- Frozen input value (sign-of-life monitoring)

### Channel error

At the output parameter `Bad`, channel errors are displayed with 1. The channel error is generated from the signal status `PV_ST`.

### Higher-level error / invalid measuring range

A higher-level error is displayed at the output parameters `ModErr` and `Bad` with 1 if the signal status at `High Word` of the input parameter `Mode` accepts the value 16#40.

A higher-level error is also present when an incorrect measuring type is entered in `Low Word` of the input parameter `Mode`.

In addition, at the output parameter `PV_Li` of the signal status, either 16#00 (in the event of an error) or 16#60 (for manipulated value (for example, substitute value, simulation, last valid value) is output.

### See also

FbAnIn block diagram (Page 2475)
FbAnIn I/Os (Page 2472)
FbAnIn messaging (Page 2472)
FbAnIn functions (Page 2468)
FbAnIn modes (Page 2468)
Description of FbAnIn (Page 2466)

## 17.2.5 FbAnIn messaging

### Messaging

This block does not offer messaging.

### See also

FbAnIn block diagram (Page 2475)
FbAnIn I/Os (Page 2472)
FbAnIn error handling (Page 2471)
FbAnIn functions (Page 2468)
FbAnIn modes (Page 2468)
Description of FbAnIn (Page 2466)

## 17.2.6 FbAnIn I/Os

I/Os of FbAnIn

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 2468) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 28: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 0<br>• 0 |
| FlutEn | 1 = Flutter suppression activated | BOOL | 0 |
| FlutTmIn* | Flutter time [s] | INT | 0 |
| FrznEn | 1 = Sign-of-life monitoring activated | BOOL | 0 |
| FrznTmIn | Monitoring time in [s] | REAL | 0.0 |
| LowCutOff | Input value Lowcutoff | REAL | 0.0 |
| MS | Maintenance status | DWORD | 16#00000000 |
| MS_Release | Release for maintenance (interconnected with MS_Release of the technology block) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ms_Ext | External maintenance status. For more information, refer to the MS_Ext (Page 2463) parameter section under information on using channel blocks. | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV | Process value (analog value) | REAL | 0.0 |
| PVSlv | Process value of the slave | REAL | 0.0 |
| PVSlv_ST | State of the slave process value | BYTE | 16#00 |
| PV_ST | Signal status for the process value | BYTE | 16#80 |
| PV_Unit | Unit of measure for process value | INT | 1001 |
| SampleTime | Sampling time in [s] | REAL | 0.1 |
| Scale | Scaling of the process value as a structure | STRUCT<br>• High: REAL<br>• Low:REAL | -<br>• 100.0<br>• 0.0 |
| SimOn | 1 = Simulation on | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimPV | Process value used for SimOn = 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SubsPV | Substitute value | REAL | 0.0 |
| TextRef | Text reference | WORD | 16#0000 |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## In/out parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| DataXchg | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |
| Mode | Value status and measuring type | DWORD | 16#00000000 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Bad | 1 = Process value is not valid | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see FbAnIn error handling (Page 2471) | INT | -1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| FrznVal | Frozen process value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ModErr | 1 = Device/module is faulty | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Field device is undergoing maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Li | Standard value (physical variable) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_LiUnit | Unit of the process value | INT | 0 |
| PV_LcoAct | Limited to within range limits Lowcutoff | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RemTime | Remaining monitoring time [s] | REAL | 0.0 |
| SampleTime | Sampling time [s] | REAL | 0.1 |
| ScaleOut | Scaling of the process value as a structure | STRUCT<br>• High: REAL<br>• Low:REAL | -<br>• 100.0<br>• 0.0 |
| SimAct | 1 = Simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |

## Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| Mode | IN / OUT | x | | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| MS_Xchg | IN / OUT | x | | |
| MS | IN | x | | |
| PV_ST | IN | | x | |
| PVSlv | IN | | x | |
| PVSlv_ST | IN | | x | |

### See also

FbAnIn block diagram (Page 2475)

FbAnIn messaging (Page 2472)

FbAnIn modes (Page 2468)

Description of FbAnIn (Page 2466)

Parameters that cannot be connected (Page 75)

## 17.2.7 FbAnIn block diagram

### FbAnIn block diagram

A block diagram is not provided for this block.

### See also

FbAnIn I/Os (Page 2472)

FbAnIn messaging (Page 2472)

FbAnIn error handling (Page 2471)

FbAnIn functions (Page 2468)

FbAnIn modes (Page 2468)

Description of FbAnIn (Page 2466)

# 17.3 FbAnOu - Analog output channel block for field devices

## 17.3.1 Description of FbAnOu

### Object name (type + number) and family

Type + number: FB 1814

Family: Channel

### Area of application for FbAnOu

The block is used for the following applications:

Signal processing (cyclic service) in accordance with "Actuator" PROFIBUS PA profile of an analog input value:

- Of a PA field device in accordance with PROFIBUS 3.0 class A and B
- Of an FF field device

### How it works

Block FbAnOu cyclically reads the process values and the signal status of the field device from the process image (partition). The process values are available as physical variables. The signal status contains information about the status of the field device.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

You must interconnect each used signal of the block with the symbols configured in HW Config or in the symbol table according to your user data configuration:

| PA device connection | Data type | I/O |
|---|---|---|
| `Rbk` | REAL | Input |
| `RCasOut` | REAL | Input |
| `PosD` | BYTE | Input |
| `SP` | REAL | Output |
| `RCasIn` | REAL | Output |

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameter `Mode` is interconnected with the corresponding output parameter `OMode_xx` of the MOD_PAL0, FF_MOD32- or MOD_PAX0 block.
- The in/out parameter `DataXchg` is interconnected with the corresponding output parameter `DataXchg_xx` of the MOD_PAL0, FF_MOD32- or MOD_PAX0 block.

- The MS parameter is interconnected to the O_MS output parameter of the diagnostics block.
- The corresponding signal status is symbolically interconnected depending on your user data configuration.

**Note**

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter Mode manually. Refer to the Mode settings for field devices (Page 2698) chapter for more on this.

For the FbAnOu block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- PID controller for PA/FF devices (PIDControl_Lean_Fb) (Page 2821)
- Control valve for PA/FF devices (ValveAnalog_Fb) (Page 2869)

## Startup characteristics

The block is executed once in OB100 at system start. The output and in/out parameters are calculated.

## Redundancy

The block monitors the values of the redundant signals (master and slave) with quality code and the value of the signal with "Good" quality code along with its quality code is transferred to the process.

## Status word allocation for Status parameter

This block does not have the Status parameter.

## See also

FbAnOu block diagram (Page 2486)

FbAnOu I/Os (Page 2481)

FbAnOu error handling (Page 2480)

FbAnOu functions (Page 2478)

FbAnOu modes (Page 2478)

FbAnOu messaging (Page 2480)

## 17.3.2 FbAnOu modes

### FbAnOu modes

This block does not have any modes.

## 17.3.3 FbAnOu functions

### Functions of FbAnOu

The functions for this block are listed below.

### Obtaining the standard value

The signals of the FF field device are read from the process image (partition) of the inputs and written to the process image (partition) of the outputs. The controlled variable `Rbk` and discrete position feedback`PosD`, are read, as well as the active reference variable `RCasOut`, together with its associated signal status `RbkST`, `PosD_ST` and `RCasOutST` are read and written to the output parameters `SP` and `RCasIn`, with the associated signal status `RbkST` and `RCasInST`.

Additional detailed device information (`Cbk0` - `Cbk2`) can be read as an option. The device information is available per bit at the block output.

The signal status `RbkST` or `RCasOutST` that comes directly from the device can have values from 16#00 – 16#FF.

The values of the signal status `PosD_ST` and `RbkST` of 16#84 - 16#87and 16#90 – 16#93of the FF field device are evaluated like the signal status 16#80 – 16#83.

### Flutter suppression

This block provides the standard function Flutter suppression for channel blocks (Page 73)

### Signal status for Fb channel blocks

The block provides the standard function Forming and outputting the signal status for channel blocks for field devices (Page 127).

The signal status of the process values (`RCasInLi` or `RbkLi`) is generated from internal events such as channel errors, higher-level errors, or simulations, as well as from the signal status `RbkST` or `RCasInST`, which comes directly from the device.

| Value | Meaning |
|---|---|
| 16#80 | Valid value |
| 16#60 | Manipulated value (for example, substitute value, simulation, last valid value) |
| 16#28 | Bad, process related |
| 16#68 | Uncertain, device related |
| 16#78 | Uncertain, process related |

| Value | Meaning |
|---|---|
| 16#A4 | Maintenance request present |
| 16#00 | Invalid value |

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

### Keep last valid raw value

The block provides the standard function Keep last valid raw value if input value is bad (Page 207).

If `Feature.Bit31` is active and the input `SP_Li` has bad quality (16#00 OR 16#28) the outputs `SP`, `SP_ST` keep its value and will not be actualized.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |
| 30 | Outputting a de-energized value for block-external simulation (Page 159) |
| 31 | Keep last valid raw value if input value is bad (Page 207) |

### Delay the output "Bad"

If input `DelTiBad` > 0, on receiving an internal bad value, delay timer will start. Timer will abort if `DelTiBad` is 0. Minimum value of `DelTiBad` is 0. Only when timer expires i.e., `RemDelTiBad` = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running `Bad.ST` will be 16#60. Only after the timer expires the actual `ST` be updated. Substitute value is updated at `PV_Out`, only after delay timer expires. When timer is running last valid value will be there at `PV_Out`. When substitute value is used, `ST` of `PV_Out` will be 16#60.

### See also

FbAnOu block diagram (Page 2486)

FbAnOu I/Os (Page 2481)

FbAnOu error handling (Page 2480)

FbAnOu modes (Page 2478)

Description of FbAnOu (Page 2476)

FbAnOu messaging (Page 2480)

## 17.3.4 FbAnOu error handling

### Error handling of FbAnOu

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Channel error
- Higher-level error
- Invalid measuring range

### Channel error

At the output parameter `Bad`, channel errors are displayed with 1.

The channel error is generated from the signal states `RbkST`, `RCasOutST`, and `PosD_ST`. In addition to this, a channel error (`Bad` = 1) also occurs if the signal status has a valid value (`RbkST` or `RCasOutST` is 16#80) and the present position of actuator `Rbk` or `RCasOut` has the value 16#7FFFFFFF (invalid).

### Higher-level error / invalid measuring range

A higher-level error is displayed at the output parameters `ModErr` and `Bad` with 1 if the signal status at `High Word` of the input parameter `Mode` accepts the value 16#40.

A higher-level error is also present when an incorrect measuring type is entered in `Low Word` of the input parameter `Mode`.

## 17.3.5 FbAnOu messaging

### Messaging

This block does not offer messaging.

### See also

Description of FbAnOu (Page 2476)

FbAnOu modes (Page 2478)

FbAnOu functions (Page 2478)

FbAnOu error handling (Page 2480)
FbAnOu I/Os (Page 2481)
FbAnOu block diagram (Page 2486)

## 17.3.6 FbAnOu I/Os

I/Os of FbAnOu

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| CbkBy0 | Additional information on the actuator status | BYTE | 16#00 |
| CbkBy1 | Additional information on the actuator status | BYTE | 16#00 |
| CbkBy2 | Additional information on the actuator status | BYTE | 16#00 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 2478) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 30: BOOL<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 0 |
| FlutEn | 1 = Flutter suppression activated | BOOL | 0 |
| FlutTmIn* | Flutter time [s] | INT | 0 |
| MS | Maintenance status | DWORD | 16#00000000 |
| MS_Release | Release for maintenance (interconnected with MS_Release of the technology block) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ms_Ext | External maintenance status. For more information, refer to the MS_Ext (Page 2463) parameter section under information on using channel blocks. | DWORD | 16#00000000 |
| PosD | Discrete position feedback (current position) of the valve:<br>0 = Not initialized<br>1 = Closed<br>2 = Opened<br>3 = Intermediate | BYTE | 16#00 |
| PosD_ST | Signal status of PosD | BYTE | 16#80 |
| PVSlv_ST | State of the slave process value | BYTE | 16#00 |
| RCasInLi | Setpoint for the remote cascade operating mode | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RCasOut | Setpoint from function block in device | REAL | 0.0 |
| RCasOutST | Signal status of RCasOut | BYTE | 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Rbk | Current position of actuator (actual value) | REAL | 0.0 |
| RbkST | Signal status of Rbk | BYTE | 16#80 |
| Scale | Scaling of the process value as structure for display | STRUCT<br>• High: REAL<br>• Low:REAL | -<br>• 100.0<br>• 0.0 |
| SP_Li | Setpoint | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_LiUnit | Unit of measure for setpoint | INT | 1342 |
| SimOn | 1 = Simulation on | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimPosD | Discrete position feedback for (output parameter PosD_Li), that is used for SimOn = 1. | BYTE | 16#00 |
| SimRCasInLi | Setpoint for the remote cascade operating mode RCasInLi, used for SimOn = 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SimRbk | Current position of the valve (actual value) Rbk, used for SimOn = 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SimSP_Li | Setpoint SP_Li, used for SimOn = 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| TextRef | Text reference | WORD | 16#0000 |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |
| SampleTime | Sample Time [s] | REAL | 0.1 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## In/out parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| DataXchg | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |
| Mode | Value status and measuring type | DWORD | 16#00000000 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Bad | 1 = Process value is not valid | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk0 | 1 = Field device in safe position | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk1 | 1 = Request for "local operation" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk2 | 1 = Device is operated locally | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk3 | 1 = Emergency operation is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk4 | 1 = Deviation in direction of movement | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk5 | 1 = Stop reached (actuator fully open) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk6 | 1 = Stop reached (actuator fully closed) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk7 | 1 = Runtime overshoot | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk8 | 1 = Actuator is opening | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk9 | 1 = Actuator is closing | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk10 | 1 = Alarm generated by any change to the static data (FB and TB) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk11 | 1 = Simulation mode | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Cbk12 | Not used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk13 | 1 = Internal control loop interrupted | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk14 | 1 = Closed-loop control inactive | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk15 | 1 = Self-test is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk16 | 1 = Stroke integral exceeded | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk17 | 1 = Additional input is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see FbAnOu error handling (Page 2480) | INT | -1 |
| ModErr | 1 = Device/module is faulty | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Field device is undergoing maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PosD_Li | Discrete position feedback (current position) of the valve:<br>0 = Not initialized<br>1 = Closed<br>2 = Opened<br>3 = Intermediate | BYTE | 16#00 |
| PosD_LiST | Signal status PosD_Li | BYTE | 16#00 |
| PosDCloseLi | 1=feedback Close | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PosDOpenLi | 1=feedback Open | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RCasIn | Setpoint for the remote cascade operating mode | REAL | 0.0 |
| RCasInST | Signal status of RCasIn | BYTE | 16#00 |
| RCasOutLi | Setpoint from function block in device | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| RbkLi | Current position of actuator (actual value) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ScaleOut | Scaling of the process value as structure for display | STRUCT<br>• High: REAL<br>• Low:REAL | -<br>• 100.0<br>• 0.0 |
| SimAct | 1 = Simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP | Setpoint | REAL | 0.0 |
| SP_ST | Signal status of setpoint | BYTE | 16#00 |
| SP_Slv | Output setpoint of the slave device | REAL | 0.0 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |
| SP_Unit | Unit of the process value (SP_LiUnit) | INT | 0 |

## Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| Mode | IN / OUT | x | | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| MS_Xchg | IN / OUT | x | | |
| MS | IN | x | | |
| PosD | IN | | x | |
| PosD_ST | IN | | x | |
| RCasOut | IN | | x | |
| RCasOutST | IN | | x | |
| Rbk | IN | | x | |
| RbkST | IN | | x | |
| CbkBy0 | IN | | x | |

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| CbkBy1 | IN | | x | |
| CbkBy2 | IN | | x | |
| SP | OUT | | x | |
| SP_ST | OUT | | x | |
| RCasIn | OUT | | x | |
| RCasInST | OUT | | x | |

### See also

Parameters that cannot be connected (Page 75)

## 17.3.7 FbAnOu block diagram

### FbAnOu block diagram

A block diagram is not provided for this block.

# 17.4 FbAnTot - Analog totalizer channel block for field devices

## 17.4.1 Description of FbAnTot

### Object name (type + number) and family

Type + number: FB 1817

Family: Channel

### Area of application for FbAnTot

The FbAnTot block processes the cyclic parameters of the "Totalizer" PA profile of a PA field device according to PROFIBUS 3.0 class A and B.

### How it works

The FbAnTot block reads the process value (PV) with the status byte (PV_ST) of the PROFIBUS field device (structure corresponds to the totalizer of the PA profiles) cyclically from the process image (partition). The process value is available as a physical variable. The status byte (PV_ST) contains information about the state of the PROFIBUS field device.

In addition to the status byte and to improve interconnectability, additional important detail information is provided at the output interface as structured variables or in the structures. These conform to the bit combinations specified in PROFIBUS 3.0 "General Requirements".

The PV_Set_Li and PV_Mode_Li input variables can be used to write to the process image (partition) as an option. For this, the values of input variables are transferred to the PV_Set and PV_Mode outputs, which are interconnected to the process image.

The block detects a higher-level error, for example, failure of a DP/PA link, via the Mode input parameter.

- If the high byte is Mode = 16#80, then the values in the process image (partition) are valid.
- If the high byte is Mode = 16#40 (value status = higher-level error, ModErr = TRUE), then the analog value is treated as invalid.

### Configuration

The block is installed in the CFC editor in a cyclic interrupt OB (OB30 to OB38). The block is installed automatically in the startup OB (OB100).

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The symbol for the quality code of the analog input channel is interconnected with the `PV_ST` input parameter and other selected options.
  The in/out parameter `Mode` is interconnected to the corresponding `OMODE_xx` output parameter of the MOD block.
- The `MS` parameter is interconnected to the `O_MS` output parameter of the diagnostics block.
- The in/out parameter `DataXchg` is interconnected to the corresponding `DataXchg` output parameter of the MOD block.

#### tartup characteristics

The block does not have any startup characteristics.

## 17.4.2 FbAnTot modes

#### FbAnTot modes

This block does not have any operating modes.

## 17.4.3 FbAnTot functions

#### FbAnTot functions

The functions for this block are listed below.

#### Obtaining the standard value

The analog value of the process image (partition) is output as a standard value at the `PV_Li` output parameter.

#### Holding the last value if raw value is invalid

If the block is to hold its last valid value when the analog value is invalid or in the initialization phase of the device, you must activate this function at the `Feature` Bit Outputting last valid value if raw value is invalid (Page 165).

#### Output substitute value if raw value is invalid

If the block is to output a substitute value (`SubsPV`) when the analog value is invalid or in the initialization phase of the device, you must activate this function at the `Feature` Bit Output substitute value if raw value is invalid (Page 160).

### Outputting an invalid value if analog value is invalid

If the block is to output an invalid value (PV_Li = PV), you must activate this function at the Feature Bit Output invalid raw value (Page 185). This function is preset.

### Flutter suppression

This block provides the standard function Flutter suppression for channel blocks (Page 73)

### Signal status for Fb channel blocks

The block provides the standard function Forming and outputting the signal status for channel blocks for field devices (Page 127).

The signal status of process value PV_Li is generated from internal events such as channel errors, higher-level errors, or simulations, as well as from the signal status PV_ST, which comes directly from the device.

The signal status PV_ST can accept values of 16#00 - 16#FF.

The block recognizes a higher-level error, for example, failure of a DP/PA link, via the Mode input parameter.

- If the high byte is Mode = 16#80, then the values in the process image (partition) are valid.
- If the high byte Mode = 16#40 (value status = higher-level error, ModErr = 1), then the analog value is treated as invalid.

The measuring type set in the low word of the Mode input parameter will be ignored.

The bit combinations of signal status PV_ST are output as output parameters (BOOL values). These conform to the bit combinations specified in PROFIBUS 3.0 "General Requirements".

**For FF field devices only:** The values of the signal status PV_ST = 16#84 - 16#87 and 16#90 – 16#93 are evaluated by the link in the same way as signal status 16#80 – 16#83.

If the signal status PV_ST = 16#80 and a process value PV with the value 16#7FFFFFFF (invalid) are transferred from the FF field device, the block will deal with the signal status 16#00 (invalid) from the FF field device.

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

### Configurable reactions using the Feature parameter

You can find an overview of all reactions provided by the Feature parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |
| 28 | Output invalid raw value (Page 185) |

| Bit | Function |
|---|---|
| 29 | Output substitute value if raw value is invalid (Page 160) |
| 30 | Outputting last valid value if raw value is invalid (Page 165) |

### Sign-of-life monitoring

If an input value whose signal status is `16#80` (good) remains constant for a programmable time (monitoring time), the input value is detected as faulty and the output parameters `Bad = 1` and `FrzVal = 1` are set.

The monitoring time is set at the `FrznTmIn` input parameter in seconds. With `FrznTmIn = 0` or `FrznEn = 0` (default setting), the sign-of-life monitoring is deactivated, any pending errors are reset.

The input value is considered as faulty as long as it seen as constant. The monitoring time is restarted each time the input value is changed.

### Delay the output "Bad"

If input `DelTiBad` > 0, on receiving an internal bad value, delay timer will start. Timer will abort if `DelTiBad` is 0. Minimum value of `DelTiBad` is 0. Only when timer expires i.e., `RemDelTiBad` = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running `Bad.ST` will be 16#60. Only after the timer expires the actual `ST` be updated. Substitute value is updated at `PV_Out`, only after delay timer expires. When timer is running last valid value will be there at `PV_Out`. When substitute value is used, `ST` of `PV_Out` will be 16#60.

## 17.4.4 FbAnTot error handling

### Error handling of FbAnTot

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. Bit is set during startup |

### See also

Error handling (Page 128)

## 17.4.5 FbAnTot messaging

### Messaging

This block does not offer messaging.

## 17.4.6 FbAnTot I/Os

I/Os of FbAnTot

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 0 |
| Feature | I/O for additional functions (Page 2488) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 28: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 0<br>• 0 |
| FlutEn | 1 = Flutter suppression activated | BOOL | 0 |
| FlutTmIn* | Flutter time [s] | INT | 0 |
| FrznEn | 1 = Sign-of-life monitoring activated | BOOL | 0 |
| FrznTmIn | Monitoring time in [s] | REAL | 0.0 |
| MS | Maintenance status | DWORD | 16#00000000 |
| MS_Release | Release for maintenance (interconnected with MS_Release of the technology block) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Ext | External maintenance status. For more information, refer to the MS_Ext (Page 2463) parameter section under information on using channel blocks. | DWORD | 16#00000000 |
| PV | Process value (analog value) | REAL | 0.0 |
| PV_Mode_Li | Mode totalizer<br>0 = Balanced,<br>1 = pos_only,<br>2 = neg_only,<br>3 = hold | BYTE | 16#00 |
| PV_Set_Li | Algorithm<br>0 = total,<br>1 = reset assign 0 ,<br>2 = Default assign PRESET_TOT | BYTE | 16#00 |
| PV_ST | Signal status for the process value | BYTE | 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_Unit | Unit of measure for process value | INT | 1001 |
| SampleTime | Sampling time in [s] | REAL | 0.1 |
| Scale | Scaling of the process value as a structure | STRUCT | - |
| | | • High: REAL | • 100.0 |
| | | • Low:REAL | • 0.0 |
| SimOn | 1 = Simulation on | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| SimPV | Process value used for SimOn = 1 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| SubsPV | Substitute value | REAL | 0.0 |
| TextRef | Text reference | WORD | 16#0000 |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## In/out parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| DataXchg | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |
| Mode | Value status and measuring type | DWORD | 16#00000000 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Bad | 1 = Process value is not valid | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see FbAnTot error handling (Page 2490) | INT | -1 |
| FrznVal | Frozen process value | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| ModErr | 1 = Device/module is faulty | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Field device is undergoing maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Li | Standard value (physical variable) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_LiUnit | Unit of the process value | INT | 0 |
| PV_Mode | Adder in operation | BYTE | 16#00 |
| PV_Set | Add up algorithmically | BYTE | 16#00 |
| RemMonTm | Remaining monitoring time | REAL | 0.0 |
| ScaleOut | Scaling of the process value as a structure | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| SimAct | 1 = Simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |

## Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| Mode | IN / OUT | x | | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| MS_Xchg | IN / OUT | x | | |
| MS | IN | x | | |
| PV_Set | OUT | | x | |
| PV_Mode | OUT | | x | |
| PV_ST | IN | | x | |

**See also**

Parameters that cannot be connected (Page 75)

## 17.4.7 FbAnTot block diagram

### FbAnTot block diagram

A block diagram is not provided for this block.

# 17.5 FbDiIn - Digital input channel block for field devices

## 17.5.1 Description of FbDiIn

### Object name (type + number) and family

Type + number: FB 1815

Family: Channel

### Area of application for FbDiIn

The block is used for the following applications:

Signal processing of digital input values (discrete input) of a field device (cyclic service in accordance with PROFIBUS PA):

- Of a PA field device in accordance with PROFIBUS 3.0 class A and B
- Of an FF field device

### How it works

Block FbDiIn cyclically reads the process values and the signal status of the field device from the process image (partition). The process values are grouped in one byte. The signal status contains information about the status of the field device.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameter Mode is interconnected with the corresponding output parameter `OMode_xx` of the MOD_PAL0, FF_MOD32- or MOD_PAX0 block.
- The in/out parameter `DataXchg` is interconnected with the corresponding output parameter `DataXchg_xx` of the MOD_PAL0, FF_MOD32- or MOD_PAX0 block.
- The symbol for the signal status of the analog input channel is interconnected to input `PV_ST`.
- The `MS` parameter is interconnected to the `O_MS` output parameter of the diagnostics block.

**Note**

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter `Mode` manually. Refer to the Mode settings for field devices (Page 2698) chapter for more on this.

For the FbDiIn block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Monitoring a digital process tag for PA/FF devices (DigitalMonitoring_Fb) (Page 2852)

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

FbDiIn block diagram (Page 2503)
FbDiIn I/Os (Page 2500)
FbDiIn messaging (Page 2500)
FbDiIn error handling (Page 2499)
FbDiIn functions (Page 2497)
FbDiIn modes (Page 2496)

## 17.5.2 FbDiIn modes

### FbDiIn modes

This block does not have any modes.

### See also

FbDiIn block diagram (Page 2503)
FbDiIn I/Os (Page 2500)
FbDiIn messaging (Page 2500)
FbDiIn error handling (Page 2499)
FbDiIn functions (Page 2497)
Description of FbDiIn (Page 2495)

## 17.5.3 FbDiIn functions

### Functions of FbDiIn

The functions for this block are listed below.

### Obtaining the standard value

The digital values (WORD format) of the process image (partition) are output at the `PV_Li0` - `PV_Li7` output parameters.

The signal status `PV_ST` that comes directly from the device can have values from 16#00 – 16#FF.

The values of the signal status `PV_ST` of 16#84 - 16#87 and 16#90 – 16#93 of the FF field device are evaluated in the same way as 16#80 – 16#83.

### Holding the last value if raw value is invalid

If the block is to hold its last valid value when the digital value is invalid or in the initialization phase of the device, you must activate this function at the `Feature` Bit Outputting last valid value if raw value is invalid (Page 165).

### Output substitute value if raw value is invalid

If the block is to output a substitute value (`SubsPV`) when the digital value is invalid or in the initialization phase of the device, you must activate this function at the `Feature` Bit Output substitute value if raw value is invalid (Page 160).

### Output of invalid value if raw value is invalid

If the block is to output an invalid value (`PV_Li` = `PV`), you must activate this function at the `Feature` Bit Output invalid raw value (Page 185).

This function is preset.

### Flutter suppression

This block provides the standard function Flutter suppression for channel blocks (Page 73)

### Signal status for Fb channel blocks

The block provides the standard function Forming and outputting the signal status for channel blocks for field devices (Page 127).

The signal status of the process value `PV_Li0` - `PV_Li7` is generated from internal events such as channel errors, higher-level errors, or simulations, as well as from the signal status `PV_ST`, which comes directly from the device.

| Value | Meaning |
|---|---|
| 16#80 | Valid value |
| 16#60 | Manipulated value (for example, substitute value, simulation, last valid value) |
| 16#28 | Bad, process related |
| 16#68 | Uncertain, device related |
| 16#78 | Uncertain, process related |
| 16#A4 | Maintenance request present |
| 16#00 | Invalid value |

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |
| 28 | Output invalid raw value (Page 185) |
| 29 | Output substitute value if raw value is invalid (Page 160) |
| 30 | Outputting last valid value if raw value is invalid (Page 165) |

### Delay the output "Bad"

If input `DelTiBad` > 0, on receiving an internal bad value, delay timer will start. Timer will abort if `DelTiBad` is 0. Minimum value of `DelTiBad` is 0. Only when timer expires i.e., `RemDelTiBad` = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running `Bad.ST` will be 16#60. Only after the timer expires the actual `ST` be updated. Substitute value is updated at `PV_Out`, only after delay timer expires. When timer is running last valid value will be there at `PV_Out`. When substitute value is used, `ST` of `PV_Out` will be 16#60.

### See also

FbDiIn block diagram (Page 2503)

FbDiIn I/Os (Page 2500)

FbDiIn messaging (Page 2500)

FbDiIn error handling (Page 2499)
FbDiIn modes (Page 2496)
Description of FbDiIn (Page 2495)

## 17.5.4 FbDiIn error handling

### Error handling of FbDiIn

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Channel error
- Higher-level error
- Invalid measuring range

### Channel error

At the output parameter `Bad`, channel errors are displayed with 1. The channel error is generated from the signal status `PV_ST`.

### Higher-level error / invalid measuring range

A higher-level error is displayed at the output parameters `ModErr` and `Bad` with 1 if the signal status at `High Word` of the input parameter `Mode` accepts the value 16#40.

A higher-level error is also present when an incorrect measuring type is entered in `Low Word` of the input parameter `Mode`.

In addition, at the output parameter `PV_LiX` of the signal status, either 16#00 (in the event of an error) or 16#60 (for manipulated value (for example, substitute value, simulation, last valid value) is output. (`PV_LiX`: X = 0 ... 7)

### See also

FbDiIn block diagram (Page 2503)
FbDiIn I/Os (Page 2500)
FbDiIn messaging (Page 2500)
FbDiIn functions (Page 2497)
FbDiIn modes (Page 2496)
Description of FbDiIn (Page 2495)

## 17.5.5 FbDiIn messaging

### Messaging

This block does not offer messaging.

### See also

FbDiIn block diagram (Page 2503)
FbDiIn I/Os (Page 2500)
FbDiIn error handling (Page 2499)
FbDiIn functions (Page 2497)
FbDiIn modes (Page 2496)
Description of FbDiIn (Page 2495)

## 17.5.6 FbDiIn I/Os

FbDiIn I/Os

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 2497) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 28: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 0<br>• 0 |
| FlutEn | 1 = Flutter suppression activated | BOOL | 0 |
| FlutTmIn* | Flutter time [s] | INT | 0 |
| MS | Maintenance status | DWORD | 16#00000000 |
| MS_Release | Release for maintenance (interconnected with MS_Release of the technology block) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Ext | External maintenance status. For more information, refer to the MS_Ext (Page 2463) parameter section under information on using channel blocks. | DWORD | 16#00000000 |
| PV | Process value (digital value) | BYTE | 16#00 |
| PV_ST | Signal status for the process value | BYTE | 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SimOn | 1 = Simulation on | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimPV0 | Process value used for SimOn = 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimPV1 | Process value used for SimOn = 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimPV2 | Process value used for SimOn = 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimPV3 | Process value used for SimOn = 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimPV4 | Process value used for SimOn = 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimPV5 | Process value used for SimOn = 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimPV6 | Process value used for SimOn = 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimPV7 | Process value used for SimOn = 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SubsPV | Substitute value | BYTE | 16#00 |
| TextRef | Text reference | WORD | 16#0000 |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |
| SampleTime | Sample Time [s] | REAL | 0.1 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## In/out parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| DataXchg | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |
| Mode | Value status and measuring type | DWORD | 16#00000000 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Bad | 1 = Process value is not valid | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see FbDiIn error handling (Page 2499) | INT | -1 |
| ModErr | 1 = Device/module is faulty | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Field device is undergoing maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Li0 | Process value 0 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Li1 | Process value 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Li2 | Process value 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Li3 | Process value 3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Li4 | Process value 4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Li5 | Process value 5 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Li6 | Process value 6 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_Li7 | Process value 7 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1 = Simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |

### Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| Mode | IN / OUT | x | | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| MS_Xchg | IN / OUT | x | | |
| MS | IN | x | | |
| PV_ST | IN | | x | |

### See also

FbDiIn block diagram (Page 2503)

FbDiIn messaging (Page 2500)

FbDiIn modes (Page 2496)

Description of FbDiIn (Page 2495)

Parameters that cannot be connected (Page 75)

## 17.5.7 FbDiIn block diagram

### FbDiIn block diagram

A block diagram is not provided for this block.

**See also**

FbDiIn I/Os (Page 2500)
FbDiIn messaging (Page 2500)
FbDiIn error handling (Page 2499)
FbDiIn modes (Page 2496)
FbDiIn functions (Page 2497)
Description of FbDiIn (Page 2495)

# 17.6 FbDiOu - Digital output channel block for field devices

## 17.6.1 Description of FbDiOu

### Object name (type + number) and family

Type + number: FB 1816

Family: Channel

### Area of application for FbDiOu

The block is used for the following applications:

Signal processing of max 8 digital input/output values of a field device (cyclic service in accordance with PROFIBUS PA):

- Of a PA field device in accordance with PROFIBUS 3.0 class A and B
- Of an FF field device

### How it works

Block FbDiOu cyclically reads the process values and the signal status of the field device from the process image (partition). The eight process values are grouped in one byte for each of the input parameters `SP_Li` and `RCasInLi`. The signal status contains information about the status of the field device.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

You must interconnect each used signal of the block with the symbols configured in HW Config or in the symbol table according to your user data configuration:

| I/O field device | Data type | I/O |
|---|---|---|
| `Rbk` | BYTE | Input |
| `RCasOut` | BYTE | Input |
| `SP` | BYTE | Output |
| `RCasIn` | BYTE | Output |

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameter Mode is interconnected with the corresponding output parameter `OMode_xx` of the MOD_PAL0, FF_MOD32- or MOD_PAX0 block.
- The in/out parameter `DataXchg` is interconnected with the corresponding output parameter `DataXchg_xx` of the MOD_PAL0, FF_MOD32- or MOD_PAX0 block.

- The MS parameter is interconnected to the O_MS output parameter of the diagnostics block.
- The corresponding signal status is symbolically interconnected depending on your user data configuration.

**Note**

If you are not using the CFC function "Generate module drivers" you must set the Mode in/out parameter manually. Refer to the Mode settings for field devices (Page 2698) section for more on this.

For the FbDiOu block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Dosing with PA/FF devices (Dose_Lean_Fb) (Page 2857)

### Startup characteristics

The block will run through one time in OB100 at system start. The output and in/out parameters are calculated.

### Status word allocation for Status parameter

This block does not have the Status parameter.

### See also

FbDiOu block diagram (Page 2515)

FbDiOu I/Os (Page 2510)

FbDiOu messaging (Page 2510)

FbDiOu error handling (Page 2509)

FbDiOu functions (Page 2507)

FbDiOu modes (Page 2506)

## 17.6.2 FbDiOu modes

### FbDiOu modes

This block does not have any modes.

### See also

FbDiOu block diagram (Page 2515)

FbDiOu I/Os (Page 2510)

FbDiOu messaging (Page 2510)

FbDiOu error handling (Page 2509)

FbDiOu functions (Page 2507)

Description of FbDiOu (Page 2505)

## 17.6.3 FbDiOu functions

### Functions of FbDiOu

The functions for this block are listed below.

### Obtaining the standard value

The signals of the FF field device are read from the process image (partition) of the inputs and written to the process image (partition) of the outputs. The input parameter `Rbk` and the active reference variable `RCasOut`, together with its associated signal status `RbkST` and `RCasOutST` are read and written to the output parameters `SP` and `RCasIn` with the associated signal status `RbkST` and `RCasOutSt`. Additional detailed device information (`CbkBy0` - `CbkBy2`) can be read as an option. The device information is available per bit at the block output.

The signal status `RbkST` or `RCasOutST` that comes directly from the device can have values from 16#00 – 16#FF.

The values of the signal status `RbkST` and `RCasOutST` of 16#84 - 16#87and 16#90 – 16#93of the FF field device are evaluated by the link as 16#80 – 16#83.

### Flutter suppression

This block provides the standard function Flutter suppression for channel blocks (Page 73)

### Signal status for Fb channel blocks

The block provides the standard function Forming and outputting the signal status for channel blocks for field devices (Page 127).

The signal status of the process values (`RCasIn` or `RbkLi`) is generated from internal events such as channel errors, higher-level errors, or simulations, as well as from the signal status `RbkLiST` or `RCasInST`, which comes directly from the device.

| Value | Meaning |
|---|---|
| 16#80 | Valid value |
| 16#60 | Manipulated value (for example, substitute value, simulation, last valid value) |
| 16#28 | Bad, process related |

| Value | Meaning |
|---|---|
| 16#68 | Uncertain, device related |
| 16#78 | Uncertain, process related |
| 16#A4 | Maintenance request present |
| 16#00 | Invalid value |

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

### Keep last valid raw value

The block provides the standard function Keep last valid raw value if input value is bad (Page 207).

If `Feature.Bit31` is active and the input `SP_Li` has bad quality (16#00 OR 16#28) the outputs `SP`, `SP_ST` keep its value and will not be actualized.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |
| 30 | Outputting a de-energized value for block-external simulation (Page 159) |
| 31 | Keep last valid raw value if input value is bad (Page 207) |

### Delay the output "Bad"

If input `DelTiBad` > 0, on receiving an internal bad value, delay timer will start. Timer will abort if `DelTiBad` is 0. Minimum value of `DelTiBad` is 0. Only when timer expires i.e., `RemDelTiBad` = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running `Bad.ST` will be 16#60. Only after the timer expires the actual `ST` be updated. Substitute value is updated at `PV_Out`, only after delay timer expires. When timer is running last valid value will be there at `PV_Out`. When substitute value is used, `ST` of `PV_Out` will be 16#60.

### See also

FbDiOu block diagram (Page 2515)

FbDiOu I/Os (Page 2510)

FbDiOu messaging (Page 2510)

FbDiOu error handling (Page 2509)
FbDiOu modes (Page 2506)
Description of FbDiOu (Page 2505)

## 17.6.4 FbDiOu error handling

### Error handling of FbDiOu

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Channel error
- Higher-level error
- Invalid measuring range

### Channel error

At the output parameter `Bad`, channel errors are displayed with 1. The channel error is generated from the signal states `RbkST` and `RCasOutSt`.

### Higher-level error / invalid measuring range

A higher-level error is displayed at the output parameters `ModErr` and `Bad` with 1 if the signal status at `High Word` of the input parameter `Mode` accepts the value 16#40.

A higher-level error is also present when an incorrect measuring type is entered in `Low Word` of the input parameter `Mode`.

### See also

FbDiOu block diagram (Page 2515)
FbDiOu I/Os (Page 2510)
FbDiOu messaging (Page 2510)
FbDiOu functions (Page 2507)
FbDiOu modes (Page 2506)
Description of FbDiOu (Page 2505)

## 17.6.5 FbDiOu messaging

### Messaging

This block does not offer messaging.

### See also

FbDiOu block diagram (Page 2515)

FbDiOu I/Os (Page 2510)

FbDiOu error handling (Page 2509)

FbDiOu functions (Page 2507)

FbDiOu modes (Page 2506)

Description of FbDiOu (Page 2505)

## 17.6.6 FbDiOu I/Os

I/Os of FbDiOu

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `CbkBy0` | Additional information on the actuator status | BYTE | 16#00 |
| `CbkBy1` | Additional information on the actuator status | BYTE | 16#00 |
| `CbkBy2` | Additional information on the actuator status | BYTE | 16#00 |
| `EN` | 1 = Called block will be processed | BOOL | 1 |
| `Feature` | I/O for additional functions (Page 2507) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 30: BOOL<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0<br>• 0 |
| `FlutEn` | 1 = Flutter suppression activated | BOOL | 0 |
| `FlutTmIn*` | Flutter time [s] | INT | 0 |
| `MS` | Maintenance status | DWORD | 16#00000000 |
| `MS_Release` | Release for maintenance (interconnected with `MS_Release` of the technology block) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `MS_Ext` | External maintenance status. For more information, refer to the `MS_Ext` (Page 2463)parameter section under information on using channel blocks. | DWORD | 16#00000000 |
| `RCasInLi` | Setpoint for the remote cascade operating mode | BYTE | 16#00 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| RCasInLiST | Signal status for the setpoint value (RCasInLi) | BYTE | 16#80 |
| Rbk | Current position of actuator (actual value) | BYTE | 16#00 |
| RbkST | Signal status of Rbk | BYTE | 16#80 |
| RCasOut | Setpoint from function block in device | BYTE | 16#00 |
| RCasOutST | Signal status of RCasOut | BYTE | 16#80 |
| SP_Li | Setpoint | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimOn | 1 = Simulation on | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimRCasInLi | Setpoint for the remote cascade operating mode RCasInLi, used for SimOn = 1 | BYTE | 16#00 |
| SimRbk | 0 = Actuator is closed<br>1 = Actuator is open | BOOL | 0 |
| SimSP_Li | 1 = Setpoint SP_Li, used for SimOn = 1 | BOOL | 0 |
| TextRef | Text reference | WORD | 16#0000 |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |
| SampleTime | Sample Time [s] | REAL | 0.1 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## In/out parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| DataXchg | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |
| Mode | Value status and measuring type | DWORD | 16#00000000 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Bad | 1 = Process value is not valid | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk0 | 1 = Field device in safe position active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Cbk1 | 1 = Request for "manual mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk2 | 1 = Field device in "manual mode" | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk3 | 1 = Emergency override active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk4 | 1 = Current position differs from expected position | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk5 | 1 = Valve connection break | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk6 | 1 = Indicates a short-circuit at the valve connection | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk7 | 1 = Not used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk8 | 1 = Actuator is opening | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk9 | 1 = Actuator is closing | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk10 | 1 = Alarm generated by any change to the static data (FB and TB) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk11 | 1 = Simulation of process values is enabled | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk12 | 1 = Not used | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk13 | 1 = Internal control loop interrupted | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Cbk14 | 1 = Valve not active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk15 | 1 = Device under self test | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk16 | 1 = Valve travel limit has been exceeded | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk17 | 1 = Limit of break time exceeded when changing from OPEN to CLOSE | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk18 | 1 = Limit of break time exceeded when changing from CLOSE to OPEN | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk19 | 1 = Error occurred in the internal cycle test | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk20 | 1 = Timeout during the transition from OPEN to CLOSE | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk21 | 1 = Timeout during the transition from CLOSE to OPEN | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk22 | 1 = Valve blocked mechanically | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Cbk23 | 1 = Zero point not reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see FbDiOu error handling (Page 2509) | INT | -1 |
| ModErr | 1 = Device/module is faulty | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OosAct | 1 = Field device is undergoing maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP_ChnST | Signal status of the output channel and value of SP_Out | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#00 |
| RCasOutLi | Setpoint from function block in device | BYTE | 16#00 |
| RCasOutLiST | Signal status for setpoint from function block in device (RCasOutLi) | BYTE | 16#00 |
| RCasIn | Setpoint for the remote cascade operating mode | BYTE | 16#00 |
| RCasInST | Signal status of RCasIn | BYTE | 16#00 |
| RbkCloseLi | 1 = Actuator closed feedback signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RbkLi | Current position of actuator (actual value) | BYTE | 16#00 |
| RbkLiST | Signal status of current position of actuator (RbkLi) | BYTE | 16#00 |
| RbkOpenLi | 1 = Actuator open feedback signal | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1 = The block is in simulation | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SP | Setpoint | BYTE | 16#00 |
| SP_ST | Signal status of setpoint | BYTE | 16#00 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |

## Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| Mode | IN / OUT | x | | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| MS_Xchg | IN / OUT | x | | |
| MS | IN | x | | |
| RCasOut | IN | | x | |
| RCasOutST | IN | | x | |
| Rbk | IN | | x | |

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| RbkST | IN | | x | |
| CbkBy0 | IN | | x | |
| CbkBy1 | IN | | x | |
| CbkBy2 | IN | | x | |
| SP | OUT | | x | |
| SP_ST | OUT | | x | |
| RCasIn | OUT | | x | |
| RCasInST | OUT | | x | |

**See also**

FbDiOu block diagram (Page 2515)

FbDiOu messaging (Page 2510)

FbDiOu modes (Page 2506)

Description of FbDiOu (Page 2505)

Parameters that cannot be connected (Page 75)

## 17.6.7 FbDiOu block diagram

### FbDiOu block diagram

A block diagram is not provided for this block.

**See also**

FbDiOu I/Os (Page 2510)

FbDiOu messaging (Page 2510)

FbDiOu error handling (Page 2509)

FbDiOu functions (Page 2507)

FbDiOu modes (Page 2506)

Description of FbDiOu (Page 2505)

# 17.7 FbDrive - Channel block for compact drives

## 17.7.1 Description of FbDrive

### Object name (type + number) and family

Type + number: FB 1905

Family: Channel

### Area of application of FbDrive

The block is used for the following applications:

- Integration of compact drives in PCS 7

### How it works

The FbDrive block integrates compact drives that meet the following conditions and that can be detected by the system. The detection depends on the modelling that is used by the manufacturer to describe the device.You must check if the following conditions are fulfilled:

- Integration via GSD description.
- Device has to be detected as Normslave.
- Device has to be of slave family = 1 (drive).
- Telegram type "1" with 2 input words and 2 output words or telegram frame type "20" with 6 input words and 2 output words.
- When using GSD / Profibus, the address range of the device with the addresses of the telegram are available either on the module level or the input and the output addresses can be distributed on slots.
- Profinet devices that are integrated via GSDML are not supported at the moment. A few exceptions exist that are based on the OM integration of these devices.
- Addresses of the telegrams have to be defined word wise in the symbol table.

### Configuration

The interconnection is made symbolic to the first input word. All other interconnections are updated automatically using driver wizard.

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

The block is also installed automatically in the startup OB (OB100).

Connect the symbol generated in HW Config (symbol table) for the input channel with the `PZDIn1` input parameter.

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The Mode in/out parameter is interconnected to the corresponding `OMODE_xx` output parameter of the MOD_DRV block.
- The in/out parameter `DataXchg` is interconnected to the corresponding `DXCHG_xx` output parameter of the MOD_DRV block.
- With telegram type "1", the inputs `PZDIn1` and `PZDIn2` are interconnected with the corresponding input words and the outputs `PZDOut1` and `PZDOut2` with the corresponding output words of the drive; with telegram type "20", the inputs `PZDIn3` - `PZDIn6` are additionally interconnected with the corresponding input words of the drive.
- The `MS` parameter is interconnected to the `O_MS` output parameter of the diagnostics block.
- The parameterization of Standard Telegram Type 1 and Standard Telegram Type 20 is supported with this block.

**Note**

"Configuration of frequency converters with the APL channel block "FbDrive" in SIMATIC PCS 7 (https://support.industry.siemens.com/cs/document/64181993/configuration-of-frequency-converters-with-the-apl-channel-block-%E2%80%9Efbdrive%E2%80%9D-in-simatic-pcs-7?dti=0&lc=en-WW)"

- The use of different channel block types for the same device type of drive or inverter is not allowed.

### Startup characteristics

The block has startup characteristics.

## 17.7.2 FbDrive modes

### Operating modes of FbDrive

This block does not have any modes.

## 17.7.3 FbDrive functions

### Functions of FbDrive

The functions for this block are listed below.

### Reading messages

You can specify the format of messages using the `Feature` bit Reading messages (Page 158).

### Transmission of messages

If the block is to send messages to the upstream diagnostics block, you must activate this function at the `Feature` bit Transmission of status information of devices (Page 186).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the Feature parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 5 | Setting the scaling for the process values (Page 159) |
| 6 | Failure handling (Page 148) |
| 7 | Behavior in the case of Bus failure (Page 208) |
| 9 | Selection of metadata and diagnostic blocks (Page 205) |
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |
| 28 | Reading messages (Page 158) |
| 29 | Transmission of status information of devices (Page 186) |
| 30 | Outputting a de-energized value for block-external simulation (Page 159) |

### Forming the control word

When `STW1`(control word) is created and `Stw1ST` is simultaneously at `16#80`, the Boolean control inputs listed below have no effect and the control word `STW1` is set at the process value output `PZDOUT1`.

When the `Stw1ST` signal status has a bad value, the data is taken from the Boolean control inputs.

| | |
|---|---|
| `On` | Control word 1 bit 0 |
| `NoCoastSt` | Control word 1 bit 1 |
| `NoQuickSt` | Control word 1 bit 2 |
| `EnOp` | Control word 1 bit 3 |
| `EnRampGen` | Control word 1 bit 4 |
| `UnfreeRamp` | Control word 1 bit 5 |
| `EnSp` | Control word 1 bit 6 |
| `Ackn` | Control word 1 bit 7 |
| `Jogg1` | Control word 1 bit 8 |
| `Jogg2` | Control word 1 bit 9 |
| `Local` | Control word 1 bit 10 |
| `InvSP` | Control word 1 bit 11 |
| `Ctrl12` | Control word 1 bit 12 |
| `Ctrl13` | Control word 1 bit 13 |
| `Ctrl14` | Control word 1 bit 14 |
| `Ctrl15` | Control word 1 bit 15 |

### Setting the limits for the process value or for the calculation limits

The limits for the process value or the calculation limits are set by the SP_LiScale input.

**Calculation:**

SpeedLi.Value := (PZD2In*((SP_LiScale.HIGH-SP_LiScale.LOW)/16384.0))+SP_LiScale.LOW;

### Delay the output "Bad"

If input `DelTiBad` > 0, on receiving an internal bad value, delay timer will start. Timer will abort if `DelTiBad` is 0. Minimum value of `DelTiBad` is 0. Only when timer expires i.e., `RemDelTiBad` = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running `Bad.ST` will be 16#60. Only after the timer expires the actual `ST` be updated. Substitute value is updated at `PV_Out`, only after delay timer expires. When timer is running last valid value will be there at `PV_Out`. When substitute value is used, `ST` of `PV_Out` will be 16#60.

## 17.7.4 FbDrive error handling

### Error handling of FbDrive

Error handling of all the blocks is described in chapter Error handling (Page 128) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The ErrorNum I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 1 | Invalid process value (output parameter Bad = 1) |

## 17.7.5 FbDrive messaging

### Messaging of FbDrive

This block does not offer messaging.

## 17.7.6 FbDrive status word

### Forming the status word

The `ZSW1` (status word), or the Boolean status outputs, are set via the process value input PZDIN1.

| | | |
|---|---|---|
| Status word 1 | Bit 0 | `RdyOn` |
| Status word 1 | Bit 1 | `RdyOp` |
| Status word 1 | Bit 2 | `OpEn` |
| Status word 1 | Bit 3 | `Fault` |
| Status word 1 | Bit 4 | `NoOff2` |
| Status word 1 | Bit 5 | `NoOff3` |
| Status word 1 | Bit 6 | `SwOn` |
| Status word 1 | Bit 7 | `Warning` |
| Status word 1 | Bit 8 | `SpeedErr` |
| Status word 1 | Bit 9 | `CtrlReq` |
| Status word 1 | Bit 10 | `F_N_Reach` |
| Status word 1 | Bit 11 | `Zsw1_11` |
| Status word 1 | Bit 12 | `Zsw1_12` |
| Status word 1 | Bit 13 | `Zsw1_13` |
| Status word 1 | Bit 14 | `Zsw1_14` |
| Status word 1 | Bit 15 | `Zsw1_15` |

## 17.7.7 FbDrive I/Os

**I/Os of FbDrive**

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `Ackn` | OR operation with control word 1, bit 7 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `Ctrl12` | Control signal 12 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `Ctrl13` | Control signal 13 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `Ctrl14` | Control signal 14 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Ctrl15 | Control signal 15 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| EnOp | Control word 1 bit 3<br>1= Operator control active<br>0 = Operator control deactivated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| EnRampGen | Control word 1 bit 4<br>1 = Ramp generation active<br>0 = Ramp generation reset | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| EnSp | Control word 1 bit 6<br>1 = Setpoint active<br>0 = Setpoint deactivated | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Feature | I/O for additional functions | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 29<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 0<br>• 0 |
| FlutEn | 1 = Flutter suppression on | BOOL | 0 |
| FlutTmIn* | Flutter time: If a certain number of status changes occur within this time, the flutter is suppressed | INT | 0 |
| InvSp | Setpoint inverted | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Jogg1 | Control word 1 bit 8 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Jogg2 | Control word 1 bit 9 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Local | Control word 1 bit 10<br>1 = Control via AS<br>0 = No control via AS | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS* | Maintenance status | DWORD | 16#00000000 |
| MS_Release | Release for maintenance (interconnected with MS_Release of the technology block) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MS_Ext | Maintenance status external. For more information, refer to the MS_Ext (Page 2463) parameter section under information on using channel blocks. | DWORD | 16#00000000 |
| NoCoastSt | Control word 1 bit 1<br>1 = Do not de-energize and coast down drive<br>0 = De-energized and coast down drive | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NoQuickSt | Control word 1 bit 2<br>1 = No rapid stop<br>0 = Rapid stop | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PZDIn1 | Input word 1 - ZSW1 | WORD | 16#0000 |
| PZDIn2 | Input word 2 - NIST_A_GLATT/ FIST_GLATT | WORD | 16#0000 |
| PZDIn3 | Input word 3 - IAIST_GLATT | WORD | 16#0000 |
| PZDIn4 | Input word 4 - ITIST_GLATT/ MIST_GLATT | WORD | 16#0000 |
| PZDIn5 | Input word 5 - PIST_GLATT | WORD | 16#0000 |
| PZDIn6 | Input word 6 - MsgNamur | WORD | 16#0000 |
| PZDIn2Unit | Unit of measure for process value | INT | 1342 |
| PZDIn2Scale | Scaling of the process value as a structure | STRUCT<br>• HIGH: REAL<br>• LOW: REAL | -<br>• 100.0<br>• 0.0 |
| PZDIn3Unit | Unit of measure for process value | INT | 1342 |
| PZDIn3Scale | Scaling of the process value as a structure | STRUCT<br>• HIGH: REAL<br>• LOW: REAL | -<br>• 100.0<br>• 0.0 |
| PZDIn4Unit | Unit of measure for process value | INT | 1342 |
| PZDIn4Scale | Scaling of the process value as a structure | STRUCT<br>• HIGH: REAL<br>• LOW: REAL | -<br>• 100.0<br>• 0.0 |
| PZDIn5Unit | Unit of measure for process value | INT | 1342 |
| PZDIn5Scale | Scaling of the process value as a structure | STRUCT<br>• HIGH: REAL<br>• LOW: REAL | -<br>• 100.0<br>• 0.0 |
| PZDIn6Unit | Unit of measure for process value | INT | 1342 |
| PZDIn6Scale | Scaling of the process value as a structure | STRUCT<br>• HIGH: REAL<br>• LOW: REAL | -<br>• 100.0<br>• 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SP_Li | Speed setpoint | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SP_LiScale | Scaling of the process value as a structure (device data) | STRUCT<br>• HIGH: REAL<br>• LOW: REAL | -<br>• 50.0<br>• 0.0 |
| Stw1 | Control word | WORD | 16#0000 |
| Stw1ST | Status control word | BYTE | 16#00 |
| On | Control word 1 bit 0<br>1 = ON<br>0 = OFF | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Telegram | Control object | INT | 1 |
| TextRef | Text reference | WORD | 16#0000 |
| UnfreeRamp | Control word 1 bit 5<br>1 = Ramp generation can be changed<br>0 = Ramp generation cannot be changed | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |
| SampleTime | Sample Time [s] | REAL | 0.1 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## In/out parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| DataXchg | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| Mode | Value status and measuring type | DWORD | 16#00000000 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Bad | 1 = Process value is not valid | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| CtrlReq | 1 = Control request | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| CurrentLi | Normal value PZD 3 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| CurrentScale | Scaling PZD 3 | STRUCT<br>• HIGH: REAL<br>• LOW: REAL | -<br>• 100.0<br>• 0.0 |
| CurrentUnit | Unit of measurement PZD 3 | INT | 0 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number | INT | -1 |
| F_N_Reach | 1 = f or n reached or exceeded<br>0 = f or n not reached | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Fault | 1 = Error is present | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FreeLi | Normal value PZD 6 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| FreeScale | Scaling PZD 6 | STRUCT<br>• HIGH: REAL<br>• LOW: REAL | -<br>• 100.0<br>• 0.0 |
| FreeUnit | Unit of measurement PZD 6 | INT | 0 |
| MsgNamur | PZD 6 in VIK-NAMUR mode | WORD | 16#0000 |
| ModErr | 1 = Device/module is faulty | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NoOff2 | 1 = De-energize and coast down not active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| NoOff3 | 1 = Rapid stop not active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Field device is undergoing maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| OpEn | 1 = Enable setpoint operation | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Power1Li | Normal value PZD 4 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Power1Scale | Scaling PZD 4 | STRUCT<br>• HIGH: REAL<br>• LOW: REAL | -<br>• 100.0<br>• 0.0 |
| Power1Unit | Unit of measurement PZD 4 | INT | 0 |
| Power2Li | Normal value PZD 5 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| Power2Scale | Scaling PZD 5 | STRUCT<br>• HIGH: REAL<br>• LOW: REAL | -<br>• 100.0<br>• 0.0 |
| Power2Unit | Unit of measurement PZD 5 | INT | 0 |
| PZDOut1 | Output word 1 - STW1 | WORD | 16#0000 |
| PZDOut2 | Output word 2 - NSOLL_A/ FSOLL | WORD | 16#0000 |
| RdyOn | 1 = Ready to switch on<br>0 = Not ready to switch on | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RdyOp | 1 = Ready for operation<br>0 = Not ready for operation | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SpeedErr | 1 = Error speed within the tolerance range<br>0= Error speed outside the tolerance range | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SpeedLi | Normal value PZD 2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SpeedScale | Scaling PZD 2 | STRUCT<br>• HIGH: REAL<br>• LOW: REAL | -<br>• 100.0<br>• 0.0 |
| SpeedUnit | Unit of measurement PZD 2 | INT | % |
| SwOn | 1 = Cannot switch on | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Warning | 1 = Warning active<br>0 = No warning active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Zsw1 | Status word | WORD | 0 |
| Zsw1_11 | Device-specific | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Zsw1_12 | Device-specific | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Zsw1_13 | Device-specific | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Zsw1_14 | Device-specific | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Zsw1_15 | Device-specific | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |

## Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| Mode | IN / OUT | x | | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| MS_Xchg | IN / OUT | x | | |
| MS | IN | x | | |
| PZDIn2 | IN | | x | |
| PZDIn3 | IN | | x | |
| PZDIn4 | IN | | x | |
| PZDIn5 | IN | | x | |
| PZDIn6 | IN | | x | |
| PZDOut1 | OUT | | x | |

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| PZDOut2 | OUT | | x | |
| Telegram | IN | | | x |

**See also**

Parameters that cannot be connected (Page 75)

## 17.7.8 FbDrive block diagram

### Block diagram of FbDrive

A block diagram is not provided for this block.

# 17.8 FbEnMe - Channel block for ET 200SP Energy Meter

## 17.8.1 Description of FbEnMe

### Object name (type + number) and family

Type + Nubmer: FB 1908

Family: Channel

### Area of application for FbEnMe

The block is used for the following application:

- Integration of energy meter in ET 200SP/ET 200SP HA with firmware V6.0 in PCS 7.

**Note**

ET 200SP HA only supports PN and ET 200SP supports both DP and PN configurations.

### How it works

The block FbEnMe integrates the above mentioned module if the following conditions are fulfilled:

- In the HW Config, user selects the dataset 253 or 252 or 251 or 250 or 159.
- Module version 32 bytes input data/12 bytes output data.
- Module version 32 bytes input data/20 bytes output data.

### Configuration

The interconnection is made symbolic to the first input word. All other interconnections are updated automatically using a wizard.

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

The block is also installed automatically in the startup OB (OB100).

Connect the symbol generated in HW Config (symbol table) for the input channel with the `PZDIn1` input parameter.

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameters `Mode`, `DataXchg`, `DataXchg1`, and `MS_Xchg` of the FbEnMe block are interconnected with the corresponding output parameters `OMODE_00`, `DXCHG_00`, `DXCHG1_00`, and `MS_XCHG_00` of the MOD_ENME block.
- The `PZD1` to `PZD32` input parameters and the `PZDOut1` to `PZDOut12/PZDOut20` (in the case of HF V6.0 Energy Meter) output parameters are interconnected with the corresponding symbols in the process image.

- The output parameters PZDOut13 ... PZDOut20 are connected (via metadata) only in the case of Energy Meters with V6.0 as these type of meters deliver upto 20 bytes output data.
- /*+The MS parameter is interconnected to the O_MS output parameter of the MOD_ENME diagnostic block.

### Startup characteristics

The block has startup characteristics.

### Status word allocation for Status parameter

This block does not have the Status parameter.

### See also

FbEnMe modes (Page 2529)
FbEnMe functions (Page 2530)
FbEnMe error handling (Page 2531)
FbEnMe messaging (Page 2532)
FbEnMe I/Os (Page 2532)
FbEnMe block diagram (Page 2539)

## 17.8.2 FbEnMe modes

### FbEnMe operating modes

This block does not have any modes.

### See also

Description of FbEnMe (Page 2528)
FbEnMe functions (Page 2530)
FbEnMe error handling (Page 2531)
FbEnMe messaging (Page 2532)
FbEnMe I/Os (Page 2532)
FbEnMe block diagram (Page 2539)

## 17.8.3 FbEnMe functions

### Functions of FbEnMe

The functions of this block are listed below.

### Transmission of messages

If the block is to send messages to the upstream diagnostics block, you must activate this function at the `Feature` bit Transmission of status information of devices (Page 186).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139). The following functionalities are available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |
| 29 | Transmission of status information of devices (Page 186) |
| 30 | Outputting a de-energized value for block-external simulation (Page 159) |

### Forming the control word

When `STW1` (control word) is created and `Stw1ST` is simultaneously set at 16#80, the Boolean control inputs listed below have no effect and the control word `STW1` is set at the process value output `PZDOUT2`.

When the `Stw1ST` signal status has a bad value, the data is taken from the Boolean control inputs.

| | |
|---|---|
| `PZDOut1` | Variant ID from the input word `PZDIn1` |
| `PZDOut2` Bit 6 | `Opn_EnCnt` |
| `PZDOut2` Bit 7 | `Rst_EnCnt` |

**Energy meter gate:**

This function has to be activated in parameter 128 first before you can use the gate. Please refer to hardware manual. If the function is activated, the parameter “Opn_EnCnt” switches the gate to open or close.

**"Reset on request":**

With a positive edge of the parameter `Rst_EnCnt` of this channel block, the values are adopted from energy meter data record 143.

**Note**

The data record 143 of the energy meter will not be written by the channel block FbEnMe to the device. This has to be set by the operator manually.

### Delay the output "Bad"

If input `DelTiBad` > 0, on receiving an internal bad value, delay timer will start. Timer will abort if `DelTiBad` is 0. Minimum value of `DelTiBad` is 0. Only when timer expires i.e., `RemDelTiBad` = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running `Bad.ST` will be 16#60. Only after the timer expires the actual `ST` be updated. Substitute value is updated at `PV_Out`, only after delay timer expires. When timer is running last valid value will be there at `PV_Out`. When substitute value is used, `ST` of `PV_Out` will be 16#60.

### See also

Description of FbEnMe (Page 2528)

FbEnMe modes (Page 2529)

FbEnMe error handling (Page 2531)

FbEnMe messaging (Page 2532)

FbEnMe I/Os (Page 2532)

FbEnMe block diagram (Page 2539)

## 17.8.4 FbEnMe error handling

### Error handling of FbEnMe

Error handling of all the blocks is described in chapter Error handling (Page 128) in the basic instructions.

The following error can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` parameter can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |

### See also

Description of FbEnMe (Page 2528)
FbEnMe modes (Page 2529)
FbEnMe functions (Page 2530)
FbEnMe messaging (Page 2532)
FbEnMe I/Os (Page 2532)
FbEnMe block diagram (Page 2539)

## 17.8.5 FbEnMe messaging

### Messaging

This block does not offer messaging.

### See also

Description of FbEnMe (Page 2528)
FbEnMe modes (Page 2529)
FbEnMe functions (Page 2530)
FbEnMe error handling (Page 2531)
FbEnMe I/Os (Page 2532)
FbEnMe block diagram (Page 2539)

## 17.8.6 FbEnMe I/Os

### I/Os of FbEnMe

## Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| PZDIn1 | Input byte 0 | BYTE | 16#00 |
| ... | ... | ... | ... |
| PZDIn32 | Input byte 31 | BYTE | 16#00 |
| MS_Release | Release for maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS | Maintenance state | DWORD | 16#00000000 |
| MS_Ext | External maintenance status. For more information, refer to the MS_Ext (Page 2463) parameter section under information on using channel blocks. | DWORD | 16#00000000 |
| TextRef | Text 1 reference external messages | WORD | 16#0000 |
| FlutEn | 1 = Flutter suppression on | BOOL | 0 |
| FlutTmIn | Flutter suppression time | INT | 0 |
| Stw1 | Control word | BYTE | 16#00 |
| Stw1ST | Control word status | BYTE | 16#00 |
| Rst_EnCnt | Reset energy meter | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Opn_EnCnt | Open energy meter gate | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Feature | I/O for additional functions (Page 2530) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 29: BOOL<br>• Bit 30: BOOL<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 1<br>• 0 |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |
| SampleTime | Sample Time [s] | REAL | 0.1 |

## In/Out parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Mode0 | Quality and mode - Phase 1 | DWORD | 16#00000000 |
| Mode1 | Quality and mode - Phase 2 | DWORD | 16#00000000 |
| Mode2 | Quality and mode - Phase 3 | DWORD | 16#00000000 |
| DataXchg | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| PZDOut1 | Output byte 0 | BYTE | 16#00 |
| ... | ... | ... | ... |
| PZDOut20 | Output byte 20 | BYTE | 16#00 |
| Bad | 1 = Process value is not valid | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ErrorNum | Error number | INT | -1 |
| ModErr | 1 = Device/module is faulty | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Variant | User data variant | INT | 0 |
| VarError | Configured user data variant is not supported | BOOL | 1 |
| CurL1Li | Current L1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| CurL1LiScale | Scale factor of current L1 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 65.535<br>• 0.0 |
| CurL1LiUnit | Unit of current L1 | INT | 1209 |
| CurL2Li | Current L2 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| CurL2LiScale | Scale factor of current L2 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 65.535<br>• 0.0 |
| CurL2LiUnit | Unit of current L2 | INT | 1209 |
| CurL3Li | Current L3 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| CurL3LiScale | Scale factor of current L3 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 65.535<br>• 0.0 |
| CurL3LiUnit | Unit of current L3 | INT | 1209 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PL1Li | Power L1 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| PL1LiScale | Scale factor of power L1 | STRUCT | - |
| | | • High: REAL | • 27648.0 |
| | | • Low: REAL | • -27648.0 |
| PL1LiUnit | Unit of power L1 | INT | 0 |
| PL2Li | Power L2 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| PL2LiScale | Scale factor of power L2 | STRUCT | - |
| | | • High: REAL | • 27648.0 |
| | | • Low: REAL | • -27648.0 |
| PL2LiUnit | Unit of power L2 | INT | 0 |
| PL3Li | Power L3 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| PL3LiScale | Scale factor of power L3 | STRUCT | - |
| | | • High: REAL | • 27648.0 |
| | | • Low: REAL | • -27648.0 |
| PL3LiUnit | Unit of power L3 | INT | 0 |
| PTotLi | Power L1L2L3 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| PTotLiScale | Scale factor of power L1L2L3 | STRUCT | - |
| | | • High: REAL | • 27648.0 |
| | | • Low: REAL | • -27648.0 |
| PTotLiUnit | Engineering units of power L1L2L3 | INT | 0 |
| ETotLi | Energy L1L2L3 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| ETotLiScale | Scale factor of energy L1L2L3 | STRUCT | - |
| | | • High: REAL | • 4294967295.0 |
| | | • Low: REAL | • 0.0 |
| ETotLiUnit | Unit of energy L1L2L3 | INT | 0 |
| PFaL1Li | Power factor L1 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| PFaL1LiScale | Scale factor of power factor L1 | STRUCT | - |
| | | • High: REAL | • 1.0 |
| | | • Low: REAL | • 0.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PFaL1LiUnit | Unit of power factor L1 | INT | 0 |
| PFaL2Li | Power factor L2 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| PFaL2LiScale | Scale factor of power factor L2 | STRUCT | - |
| | | • High: REAL | • 1.0 |
| | | • Low: REAL | • 0.0 |
| PFaL2LiUnit | Engineering units of power factor L2 | INT | 0 |
| PFaL3Li | Power factor L3 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| PFaL3LiScale | Scale factor of power factor L3 | STRUCT | - |
| | | • High: REAL | • 1.0 |
| | | • Low: REAL | • 0.0 |
| PFaL3LiUnit | Unit of power factor L3 | INT | 0 |
| PFaTotLi | Power factor L1L2L3 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| PFaTotLiScale | Scale factor of power factor L1L2L3 | STRUCT | - |
| | | • High: REAL | • 1.0 |
| | | • Low: REAL | • 0.0 |
| PFaTotLiUnit | Unit of power factor L1L2L3 | INT | 0 |
| VL1Li | Voltage L1 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| VL1LiScale | Scale factor of voltage L1 | STRUCT | - |
| | | • High: REAL | • 300.0 |
| | | • Low: REAL | • 0.0 |
| VL1LiUnit | Engineering units of voltage L1 | INT | 1240 |
| VL2Li | Voltage L2 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| VL2LiScale | Scale factor of voltage L2 | STRUCT | - |
| | | • High: REAL | • 300.0 |
| | | • Low: REAL | • 0.0 |
| VL2LiUnit | Unit of voltage L2 | INT | 1240 |
| VL3Li | Voltage L3 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| VL3LiScale | Scale factor of voltage L3 | STRUCT | - |
| | | • High: REAL | • 300.0 |
| | | • Low: REAL | • 0.0 |
| VL3LiUnit | Unit of voltage L3 | INT | 1240 |
| VL1L2Li | Voltage L1L2 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| VL1L2LiScale | Scale factor of voltage L1L2 | STRUCT | - |
| | | • High: REAL | • 600.0 |
| | | • Low: REAL | • 0.0 |
| VL1L2LiUnit | Unit of voltage L1L2 | INT | 1240 |
| VL2L3Li | Voltage L2L3 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| VL2L3LiScale | Scale factor of voltage L2L3 | STRUCT | - |
| | | • High: REAL | • 600.0 |
| | | • Low: REAL | • 0.0 |
| VL2L3LiUnit | Unit of voltage L2L3 | INT | 1240 |
| VL3L1Li | Voltage L3L1 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| VL3L1LiScale | Scale factor of voltage L3L1 | STRUCT | - |
| | | • High: REAL | • 600.0 |
| | | • Low: REAL | • 0.0 |
| VL3L1LiUnit | Unit of voltage L3L1 | INT | 1240 |
| AcPwrLi | Active power L1 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| AcPwrLiScale | Scale factor of active power L1 | STRUCT | - |
| | | • High: REAL | • 27648.0 |
| | | • Low: REAL | • -27648.0 |
| AcPwrLiUnit | Unit of active power L1 | INT | 1186 |
| RePwrLi | Reactive power L1 | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| RePwrLiScale | Scale factor of reactive power L1 | STRUCT | - |
| | | • High: REAL | • 27648.0 |
| | | • Low: REAL | • -27648.0 |
| RePwrLiUnit | Engineering units of reactive power L1 | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ApPwrLi | Apparent power L1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ApPwrLiScale | Scale factor of apparent power L1 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 27648.0<br>• -27648.0 |
| ApPwrLiUnit | Unit of apparent power L1 | INT | 0 |
| AcErgLi | Active energy L1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| AcErgLiScale | Scale factor of active energy L1 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 4294967295.0<br>• 0.0 |
| AcErgLiUnit | Unit of active energy L1 | INT | 1175 |
| ReErgLi | Reactive energy L1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ReErgLiScale | Scale factor of reactive energy L1 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 4294967295.0<br>• 0.0 |
| ReErgLiUnit | Unit of reactive energy L1 | INT | 0 |
| ApErgLi | Apparent energy L1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| ApErgLiScale | Scale factor of apparent energy L1 | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 4294967295.0<br>• 0.0 |
| ApErgLiUnit | Unit of apparent energy L1 | INT | 0 |
| OosAct | 1 = Field device is undergoing maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OMS_Ext | Reserved | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |

### Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| Mode0 | IN / OUT | x | | |
| Mode1 | IN / OUT | x | | |
| Mode2 | IN / OUT | x | | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| MS_Xchg | IN / OUT | x | | |
| MS | IN | x | | |
| PZDInXX (XX=1 to 32) | IN | | x | |
| PZDOutXX (XX= 1 to 20) | OUT | | x | |

### See also

Description of FbEnMe (Page 2528)
FbEnMe modes (Page 2529)
FbEnMe error handling (Page 2531)
FbEnMe messaging (Page 2532)
FbEnMe block diagram (Page 2539)
Parameters that cannot be connected (Page 75)

## 17.8.7 FbEnMe block diagram

### Block diagram of FbEnMe

A block diagram is not provided for this block.

### See also

Description of FbEnMe (Page 2528)
FbEnMe modes (Page 2529)
FbEnMe functions (Page 2530)
FbEnMe error handling (Page 2531)

FbEnMe messaging (Page 2532)
FbEnMe I/Os (Page 2532)

# 17.9 FbSwtMMS - Channel block for MM starter

## 17.9.1 Description of FbSwtMMS

### Object name (type + number) and family

Type + number: FB 1907

Family: Channel

### Area of application of FbSwtMMS

The block is used for the following applications:

- Signal processing of motor management starter with profile type 1 with 2 input words and 1 output word. The Input adress of the input words must be configured.

| NOTICE |
| --- |
| **Integration of the block into technological charts**<br>Make sure when installing the block into the technological charts that the device is a switching device of slave family 2 (switching device).<br>Make sure that the device is installed according to profile type 1 for motor managed starters of the LVSG; otherwise, the block will not work correctly. The block is automatically connected with the process image according to the specifications of profile type 1. |

### How it works

The FbSwtMMS block integrates a motor management starter from any switch or starter object.

### Configuration

The interconnection is made symbolic to the first input or output word. All other interconnections are updated automatically using a wizard.

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

The block is also installed automatically in the startup OB (OB100).

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The `MODE` input is interconnected to the relevant `OMODE` output of the MOD_SWT block.
- The `PZDIn2` input and the `PZDOut1` output are interconnected using the icons of the compact drives.
- The `DataXchg` input is interconnected to the relevant `DXCHG_00` output of the MOD_SWT block.
- The `MS` input is interconnected to the relevant `O_MS` output of the MOD_SWT block.

**Note**

If motor starters which support hardware interrupts are used and if only the APL channel block `FbSwtMMS` is used, the associated hardware interrupts such as OB40 should be deactivated for these devices in order to avoid error messages and unnecessary CPU load.

"Configuration of direct starters with the APL channel block "FbSwtMMS" in SIMATIC PCS 7 (https://support.industry.siemens.com/cs/document/64182525/configuration-of-direct-starters-with-the-apl-channel-block-%E2%80%9Efbswtmms%E2%80%9D-in-simatic-pcs-7?dti=0&lc=en-DE)"

### Startup characteristics

The block has startup characteristics.

## 17.9.2 FbSwtMMS modes

### Operating modes of FbSwtMMS

This block does not have any modes.

## 17.9.3 FbSwtMMS functions

### Functions of FbSwtMMS

The functions for this block are listed below.

### Transmission of messages

If the block is to send messages to the upstream diagnostics block, you must activate this function at the `Feature` bit Transmission of status information of devices (Page 186).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the chapter Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 7 | Behavior in the case of Bus failure (Page 208) |
| 9 | Selection of metadata and diagnostic blocks (Page 205) |
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |

| Bit | Function |
|---|---|
| 29 | Transmission of status information of devices (Page 186) |
| 30 | Outputting a de-energized value for block-external simulation (Page 159) |

### Forming the control word

When STW1(control word) is created and Stw1ST is at 16#80, the Boolean control inputs listed below have no effect and the control word STW1 is set at the process value output PZDOUT1.

When the Stw1ST signal status has a bad value, the data is taken from the Boolean control inputs.

| | |
|---|---|
| Rev | Control word 1 bit 0 |
| Off | Control word 1 bit 1 |
| Fwd | Control word 1 bit 2 |
| StartTest | Control word 1 bit 3 |
| StartEmerg | Control word 1 bit 4 |
| Auto | Control word 1 bit 5 |
| ResetTrip | Control word 1 bit 6 |
| Ctrl7 | Control word 1 bit 7 |
| Ctrl8 | Control word 1 bit 8 |
| Ctrl9 | Control word 1 bit 9 |
| Ctrl10 | Control word 1 bit 10 |
| Ctrl11 | Control word 1 bit 11 |
| ManSpec1 | Control word 1 bit 12 |
| ManSpec2 | Control word 1 bit 13 |
| ManSpec3 | Control word 1 bit 14 |
| ManSpec4 | Control word 1 bit 15 |

### Delay the output "Bad"

If input DelTiBad > 0, on receiving an internal bad value, delay timer will start. Timer will abort if DelTiBad is 0. Minimum value of DelTiBad is 0. Only when timer expires i.e., RemDelTiBad = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running Bad.ST will be 16#60. Only after the timer expires the actual ST be updated. Substitute value is updated at PV_Out, only after delay timer expires. When timer is running last valid value will be there at PV_Out. When substitute value is used, ST of PV_Out will be 16#60.

## 17.9.4 FbSwtMMS error handling

### Error handling of FbSwtMMS

Error handling of all the blocks is described in chapter Error handling (Page 128) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

**Note**

If an error occurs during switching, all outputs are written as before to the field device. The inputs of the field device are displayed and the status with the value 16#00 is formed according to the MS.

**Note**

The outputs are locked during Drive, only the Local and Reset are transmitted to the device. The inputs react as with FbSwtMMS.

### Overview of error numbers

The ErrorNum I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not processed. |
| 0 | There is no error. |
| 1 | Invalid process value (output parameter Bad = 1) |

## 17.9.5 FbSwtMMS messaging

### Messaging

This block does not offer messaging.

## 17.9.6 FbSwtMMS status word

### Forming the status word

The `ZSW1` (status word), or the Boolean status outputs, is set via the process value input `PZDIN1`.

| | | |
|---|---|---|
| Status word 1 | Bit 0 | `FdkRev` |
| Status word 1 | Bit 1 | `FdkOff` |
| Status word 1 | Bit 2 | `FdkFwd` |
| Status word 1 | Bit 3 | `OverlAct` |
| Status word 1 | Bit 4 | `LockTmAct` |
| Status word 1 | Bit 5 | `FdkAuto` |

| | | |
|---|---|---|
| Status word 1 | Bit 6 | ErrAct |
| Status word 1 | Bit 7 | WarnAct |
| Status word 1 | Bit 8 | Status8 |
| Status word 1 | Bit 9 | Status9 |
| Status word 1 | Bit 10 | ManSpcO5 |
| Status word 1 | Bit 11 | ManSpcO6 |
| Status word 1 | Bit 12 | ManSpcO1 |
| Status word 1 | Bit 13 | ManSpcO2 |
| Status word 1 | Bit 14 | ManSpcO3 |
| Status word 1 | Bit 15 | ManSpcO4 |

## 17.9.7 FbSwtMMS I/Os

I/Os of FbSwtMMS

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Auto | Control command for remote control I bit 0.5 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl7 | Control signal 7 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl8 | Control signal 8 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl9 | Control signal 9 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl10 | Control signal 10 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ctrl11 | Control signal 11 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| EN | 1 = Called block will be processed | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Feature | I/O for additional functions | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 29<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 0<br>• 0 |
| FlutEn | 1 = Flutter suppression activated | BOOL | 0 |
| FlutTmln* | Flutter time: If a certain number of state changes occur within this time, fluttering is suppressed. | INT | 0 |
| Fwd | Control forward<br>I bit 0.2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| IScale | Scaling for measured current | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| IUnit | Unit of measurement for measured current | INT | 1342 |
| ManSpec1 | Device-specific command, functionality in accordance with manufacturer specifications 1<br>I bit 1.4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManSpec2 | Device-specific command, functionality in accordance with manufacturer specifications 2<br>I bit 1.5 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManSpec3 | Device-specific command, functionality in accordance with manufacturer specifications 3<br>I bit 1.6 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManSpec4 | Device-specific command, functionality in accordance with manufacturer specifications 4<br>I bit 1.7 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS* | Maintenance status | DWORD | 16#00000000 |
| MS_Release | Release for maintenance (interconnected with MS_Release of the technology block) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Ext | Maintenance status external. For more information, refer to the MS_Ext (Page 2463) parameter section under information on using channel blocks. | DWORD | 16#00000000 |
| Off | Control Off<br>I bit 0.1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PZDIn1 | Connectable input word 1 | WORD | 16#0000 |
| PZDIn2 | Connectable input word 2 | WORD | 16#0000 |
| ResetTrip | Undo last operation<br>I bit 0.6 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Rev | Control backward<br>I bit 0.0 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StartEmerg | Start command is issued despite a pending internal error<br>I bit 0.4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StartTest | Start command for an internal self-test<br>I bit 0.3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Stw1 | Control word | WORD | 16#0000 |
| Swt1ST | Status control word | BYTE | 16#00 |
| TextRef | Text reference | WORD | 16#0000 |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |
| SampleTime | Sample Time [s] | REAL | 0.1 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## In/out parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| DataXchg | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| Mode | Value status and measuring type | DWORD | 16#00000000 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Bad | 1 = Process value is not valid | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ErrAct | Status information internal device fault<br>Q bit 0.6 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ErrorNum | Output of pending error number | INT | -1 |
| FdkAuto | Feedback information for the remote control status<br>Q bit 0.5 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FdkFwd | Feedback information forward<br>Q bit 0.2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FdkOff | Feedback information Off<br>Q bit 0.1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| FdkRev | Feedback information backward<br>Q bit 0.0 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Imax | Value for highest measured current | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| IScaleOut | Scaling for measured current | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| IUnitOut | Unit of measurement for measured current | INT | 0 |
| LockTmAct | Status information, the drive is temporarily locked<br>Q bit 0.4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManSpcO1 | Device-specific feedback, functionality in accordance with manufacturer specifications 1<br>Q bit 1.4 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManSpcO2 | Device-specific feedback, functionality in accordance with manufacturer specifications 2<br>Q bit 1.5 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManSpcO3 | Device-specific feedback, functionality in accordance with manufacturer specifications 3<br>Q bit 1.6 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManSpcO4 | Device-specific feedback, functionality in accordance with manufacturer specifications 4<br>Q bit 1.7 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ManSpcO5 | Device-specific feedback, functionality in accordance with manufacturer specifications 5<br>Q bit 1.2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ManSpcO6 | Device-specific feedback, functionality in accordance with manufacturer specifications 6<br>Q bit 1.3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ModErr | 1 = Device/module is faulty | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Field device is undergoing maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OverIAct | Status information internal warning overload active<br>Q bit 0.3 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PZDOut1 | Connectable output word | WORD | 16#0000 |
| Status8 | Status output 8<br>Q bit 1.0 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Status9 | Status output 9<br>Q bit 1.1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| WarnAct | Status information internal maintenance active<br>Q bit 0.7 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Zsw1 | Status word | WORD | 16#0000 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |

### Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| Mode | IN / OUT | x | | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| MS_Xchg | IN / OUT | x | | |
| MS | IN | x | | |
| PZDIn2 | IN | | x | |
| PZDOut1 | OUT | | x | |

### See also

Parameters that cannot be connected (Page 75)

## 17.9.8 FbSwtMMS block diagram

### Block diagram of FbSwtMMS

A block diagram is not provided for this block.

# 17.10 Pcs7AITC - Analog input driver, dynamic measuring range center

## 17.10.1 Pcs7AITC description

### Object name (type + number) and family

Type + number: FB 1933

Family: Channel

### Area of application for Pcs7AITC

The block is used for the following applications:

- Signal processing of ET 200SP HA AI16 x TC/8xRTD HA (6DL1 134-6JH00-0PH1) with process values in REAL format for Measuring Type Thermocouple and Measuring range Type K [NiCr-NiAl], Type B [PtRh-PtRh], Type N [NiCrSi-NiSi], and Type S [PtRh-Pt]. It is also used in combination with I/O redundancy.
- Signal processing of ET 200SP AI4 x RTD/TC HF (6ES7 134-6JD00-0CA1) module with dynamic scalable measuring range for Measuring Type Thermocouple and Measuring Range Type K [NiCr-NiAl], Type B [PtRh-PtRh], Type N [NiCrSi-NiSi], and Type S [PtRh-Pt].

To use the mentioned modules without process values in REAL format or without scalable measuring range, you can use the channel block Pcs7AnIn.

### How it works

The block cyclically processes all the channel-specific signal functions of an analog input module. It reads the data from the process image (partition) and provides the physical value. The Input parameters that are parameterized by the driver wizard define the channel settings of the module and the corresponding behavior of Pcs7AITC.

The information exchange parameter `MrcXchg` adjusts the current Measuring Range Center (MRC) of ET 200SP analog input module. It is connected to the coordination function block WRDS_CO of the Basis Library by driver wizard.

The block supports the hardware settings for Degrees Celsius (°C) and Degrees Fahrenheit (°F).

### Configuration

You can use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). Additionally, the block is installed automatically in the startup OB (OB100).

Depending on the channel selected, the symbol generated in the HW Config (Symbol table) for the input channel can be manually connected with the input parameter `PV_InR` or `PV_In`.

**Note**

- `PV_InR` is used for AI channels with process values in REAL format for example: ET 200SP HA I/O modules.
- `PV_In` is used for AI channels with scalable measuring range for example: ET 200SP I/O modules.

In case of redundant I/O configuration, use the symbol of the channel in the master module.

When the CFC function "Generate module drivers" is used, the following operations occur automatically:

- The IN_OUT parameter `Mode` is interconnected to the corresponding `OMODE_xx` (xx = channel number) output parameter of the MOD block.
- The IN_OUT parameter `DataXchg` is interconnected to the corresponding `DXCHG_xx` (xx = channel number) output parameter of the MOD block.
- The IN_OUT parameter `DataXchg1` is interconnected to the corresponding `DXCHG1_xx` (xx = channel number) output parameter of the MOD block.
- The `MS` parameter is interconnected to the `O_MS` output parameter of the diagnostics block.
- The IN parameter `PV_sel` is set as follows:
  - `PV_sel` = 0 if the input channel uses the data type WORD for process values
  - `PV_sel` = 1 if the input channel uses the data type REAL for process values
- The Input parameter `ChValueAct` is parameterized with the channel number (0...31) from module.

If a Quality Information (QI) from the ET 200SP HA module is provided:

- the input parameter `ProImQB` is interconnected with the corresponding symbol and the input parameter `SelQB` is set to 1 (otherwise 0).

In case of I/O redundancy (ET 200SP HA):

- the interconnection with the symbols of both the input parameters `ProImQB` and `ProImQBSlv` is done.
- the interconnection with the symbol of the slave channel for input parameter `PV_InRSlv` is done.

If ET 200SP is used (only for scalable measuring range):

- the input parameter `MrcXchg` of type ANY is interconnected with output parameter `MRC_XCHG` of corresponding coordination block WRDS_CO.
- the input parameter `Scale` (`Scale.High` and `Scale.Low`) is parameterized with the start values of scaling, depending on HW Config (HWC) settings (minimum and maximum).
- the input parameter `MrcHWC` is parameterized with the start value for measuring range center from HW Config (HWC).
- `FeatureBit6` is parameterized with the information "Scalable measuring range".

- `FeatureBit7` is parameterized with the information "Scalable measuring range 2 decimal points".
- `FeatureBit8` is parameterized with the information "Scalable measuring range 3 decimal points".

**Note**

If you are not using the CFC function "Generate module drivers" you must set the `IN_OUT` parameter Mode manually. Refer to the Mode Settings for SM Modules (Page 2687) section for more on this.

The handling of channel block Pcs7AITC is similar to Pcs7AnIn thus, the Advanced Process Library does not contain own templates.

Examples of process tag types:

- Monitoring an analog process tag (AnalogMonitoring) (Page 2854)
- Cascade control with control loop monitoring through ConPerMon (CascadeControl) (Page 2840)
- Cascade control with PIDConR (CascadeR) (Page 2843)
- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 2827)
- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 2825)
- Override control with PIDConR (OverrideR) (Page 2848)
- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 2822)
- PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 2823)
- Ratio control with control loop monitoring through ConPerMon (RatioControl) (Page 2836)
- Ratio control with PIDConR (RatioR) (Page 2839)
- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 2832)
- Step controller with direct access to the actuator and without position feedback (StepControlDirect) (Page 2831)
- Model-based predictive control (ModPreCon) (Page 2849)
- Reversible motor with controllable speed (MotorSpeedControlled) (Page 2862)
- Dosing (Dose_Lean) (Page 2856)
- Split-range controller with control loop monitoring through ConPerMon (SplitrangeControl) (Page 2833)

## Startup characteristics

The accept value delay is started when `CountLim` ≠ 0.

### Quality code (ET200 SP HA modules)

The quality code is read from the process image and has two states: "Good" (true) or "Bad" (false).

### Redundancy

Depending on the QI, the function block decides which of the two redundant process values is transferred and processed.

- In case the master channel delivers a "good" state in QI, then the master channel process value is used.
- In case the master channel delivers a "bad" state and the slave channel delivers a "good" state, then the slave channel process value is used.
- In case both QI is "bad", it depends on your parameterization if master process value or a substitute value is transferred.

### Status word allocation for Status parameter

This block does not have the `Status` parameter.

### See also

Override control (Page 2846)

## 17.10.2 Pcs7AITC functions

### Functions of Pcs7AITC

The functions for this block are listed below.

### Checking the raw value

The nominal range for converting analog signals into digital values (raw values) on the module, depends on the measuring type and the range of the analog input channel. The nominal range is defined in the hardware configuration and is automatically saved in the in/out parameter when the block driver `Mode` is created.

This includes an overshoot/undershoot range within which an analog signal can still be converted to a digital value. This range is defined in relation to the nominal range (approximately 18.5%). Outside this range, an overflow or underflow occurs and the output parameter `Bad` = 1 is set.

- Output parameter `PV_LoAct` = 1 is set if the value is outside the nominal low range.
- Output parameter `PV_HiAct` = 1 is set if the value is outside the nominal high range.

If dynamic scalable measuring range of ET 200SP AI module is used, the parameters `PV_LoAct` and `PV_HiAct` are only set in case of underflow or overflow.

### Obtaining the standard value

The standard value ( temperature - a physical quantity) is obtained from the scalable measuring range of the ET 200SP AI module. The driver wizard parameterizes the input parameter `Scale` and uses it for managing the functionality of scalable measuring range.

If REAL data format is used with Thermocouple Type K, Type B, Type N, or Type S, then set the values `Scale.Low` = 0 and `Scale.High` = 1. This is because, the raw value from the module is already a physical quantity (temperature).

### Holding the last value if raw value is invalid

If the block is to hold the most recent valid value when the raw value is invalid, you must activate this function at the `Feature` Bit Outputting last valid value if raw value is invalid (Page 165).

You can also influence this function via the input parameter `DeltaVal` as follows:

- `DeltaVal` ≤ 0: the last value is retained and is not influenced
- `DeltaVal` > 0: the last or the next to last value is output

If you set the parameter `DeltaVal` > 0, the last `PV_Out`(k - 1) or next to last `PV_Out` (k - 2) is the valid output value and the (`PV_Out`(k) is the current value, where k is the current time).

At parameter `DeltaVal` you can preset a permitted process value change (`PV_Out`) between two calls.

You have the following options:

- For invalid raw values and `DeltaVal` > 0:
  - If |`PV_Out`(k - 1) - `PV_Out`(k - 2)| > `DeltaVal`, then `PV_Out` = `PV_Out`(k - 2) (last but one valid output value is the output)
  - If |`PV_Out`(k) - `PV_Out`(k - 1)| ≤ `DeltaVal`, then `PV_Out` = `PV_Out`(k - 1) (last valid output value is the output)
- For valid raw values and `DeltaVal` > 0:
  - |`PV_Out`(k) - `PV_Out`(k - 1)| > `DeltaVal`, so for one cycle `PV_Out` = `PV_Out`(k - 1) is output, that is, the input parameter. `DeltaVal` is used to limit the change made to the valid raw value. In addition, the signal status at the output parameter `PV_Out` is set to 16#60 and the output parameter is set to `Bad` = 0.
    The value of `DeltaVal` should be selected with due care. If the value is too low, the quality code may flutter between16#80 and 16#60, regardless of the status of the raw value.

### Output substitute value if raw value is invalid

If the block is to output a substitute value (`SubsPV_In`) when the raw value is invalid, you must activate the preset function at the `Feature` Bit Output substitute value if raw value is invalid (Page 160).

### Value application delay

After a restart, or if the output parameter `Bad` changes its value from 1 to 0, the signal status and the value of output parameter `PV_Out` are not updated until the number of cycles for delayed application of the value (input parameter `CountLim`) have elapsed. During the value application delay, the signal status at the output parameter is `PV_Out` = 16#00 and `Bad` = 1. The last value is retained during the value application delay.

If `CountLim` = 0 , the function is deactivated.

### Flutter suppression

This block provides the standard function Flutter suppression for channel blocks (Page 73)

### Signal status for PCS 7 channel blocks

The block provides the standard function Forming and outputting the signal status for PCS 7 channel blocks (Page 127).

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

### Configurable reactions using the Feature parameter

You can find an overview of all reactions provided by the Feature parameter in the Configurable response using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 6 | Scalable measuring range (driver wizard) (Page 203) |
| 7 | 2 decimal points (scalable measuring range) - parameterized by the driver wizard (Page 203) |
| 8 | 3 decimal points (scalable measuring range) - parameterized by the driver wizard (Page 204) |
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 11 | Reserved |
| 12 | 0 = manual new MRC value,<br>1 = AUTO trigger for new Measuring Range Center |
| 13 | 1 = AUTO value for new Measuring Range Center |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |
| 28 | Output invalid raw value (Page 185) |
| 29 | Output substitute value if raw value is invalid (Page 160) |
| 30 | Outputting last valid value if raw value is invalid (Page 165) |

### Sign-of-life monitoring

If an analog input value with signal status 16#80 (good) remains constant for a programmable time (monitoring time), the input value is detected as faulty and the output parameters `Bad` and `FrzVal` are set to 1. The signal status is set to `PV_Out.ST` = 16#00.

The monitoring time is defined at the input parameter `FrznTmIn` (in seconds). With `FrznTmIn` = 0 or `FrznEn` = 0 (default setting), the sign-of-life monitoring is deactivated, any pending errors are reset.

The input value is considered as faulty after monitoring time as long as it stays constant. The monitoring time is restarted each time the input value changes.

### Scalable measuring range (only ET 200SP)

Scalable measuring range is a functionality provided by ET 200SP AI modules. It allows an increased resolution in a reduced measuring range . The Measuring Range Center (MRC) and the resolution (2 or 3 decimal places) can be configured.

For more information, refer to the manual "*SIMATIC ET 200SP analog input module AI 4xRTD/TC 2-/3-/4-wire HF*".

Together with coordination block WRDS_CO from the Basis Library, the Pcs7AITC block can request new values for MRC, manually or automatically via WRREC of DS128 to the module. This depends on the setting of `Feature.Bit12` and `Feature.Bit13`. With these two bits the following modes are possible:

- `Feature.Bit12` = 0, `Feature.Bit13` = 0 - "manual mode with trigger input bit"
  With the input parameter `MrcVal`, a new MRC can be defined. With a rising edge on input parameter `MrcReq` the new MRC value is requested.The `MrcReq` is then reset.
- `Feature.Bit12` = 1, `Feature.Bit13` = 0 - "manual mode without trigger input bit"
  With the input parameter `MrcVal`, a new MRC can be defined. With a value change on this parameter, the new value is requested automatically.
- `Feature.Bit12` = 0, `Feature.Bit13` = 1 - "manual update of MRC "
  With a rising at the input parameter `MrcReq`, the current process value (`PV_Out.Value`) is taken for requesting a new MRC.
- `Feature.Bit12` = 1, `Feature.Bit13` = 1 - "automatic update of MRC"
  As soon as the process value approaches current `Scale` values, a new MRC is requested automatically. The corresponding percentage values that define the ranges in which a new MRC is requested, can be defined via the input parameters `MrcPL` and `MrcPH`.

With the input parameter `MrcTime`, a waiting time for a new MRC request can be defined. After having requested a new MRC, this time starts to run and within this time, no new MRC can be requested again.

The input parameter `MrcFrz` defines the time, as long as no new `PV_Out` is calculated after having requested a new MRC. It is suggested to select a bigger value for `MrcTime` than for `MrcFrz`.

The output parameter `MrcCurr` always presents the currently used MRC value. `ScaleOut` is a structured output parameter that contains the currently used `Scale` values. The output parameter `MrcBusy` displays that a new MRC was requested and that the handling is still active. The output parameter `MrcTiLo` = 1 means that the time lock is not yet over (see input parameter `MrcTime`).

If an underflow (PV_In =16#8000) or overflow (PV_In = 16#7FFF) occurs, Pcs7AITC tries to let the underflow/overflow situation disappear. If Feature.Bit12 and Feature.Bit13 are set, Pcs7AITC requests a new MRC value that is identical to the current value of ScaleOut.Low (in case of underflow) or ScaleOut.High (in case of overflow). In case Feature.Bit12 is not set, but Feature.Bit13 is set, MrcReq = 1 is necessary to start each MRC request.

This leads to the fact that in case of wire break, when the module delivers PV_In = 16#7FFF, the described mechanism starts and the output parameter MrcCurr together with Scale.Out reaches highest range. After wire break is reduced, Pcs7AITC finds a suitable MRC for the current process value. During this search that starts at the highest range, a temporary underflow (PV_In = 16#8000) occurs. This can also lead to a corresponding diagnostic from module that leads to the message "Lower limit violated" C/G of the MOD function block.

### Delay the output "Bad"

If input DelTiBad > 0, on receiving an internal bad value, delay timer will start. Timer will abort if DelTiBad is 0. Minimum value of DelTiBad is 0. Only when timer expires i.e., RemDelTiBad = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running Bad.ST will be 16#60. Only after the timer expires the actual ST be updated. Substitute value is updated at PV_Out, only after delay timer expires. When timer is running last valid value will be there at PV_Out. When substitute value is used, ST of PV_Out will be 16#60.

## 17.10.3 Pcs7AITC error handling

### Error handling of Pcs7AITC

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Channel error
- Higher-level error
- Invalid measuring range
- Frozen input value (sign-of-life monitoring)

### Channel error

At the output parameter Bad, channel errors are displayed with the value 1. Channel errors can be detected using the raw value, the Quality Information or the sign-of-life monitoring.

The output parameters PV_LoAct or PV_HiAct remain set to 1, if a channel error occurs due to the module diagnosis "Undershoot or overshoot of the measurement range".

### Higher-level error / invalid measuring range

A higher-level error output is obtained (output parameter `ModErr` = 1 and `Bad` = 1) if either:

- the signal status in the High Word of input parameter mode takes the value 16#40, or
- there is an invalid measuring type in the Low Word of the input parameter mode.

## 17.10.4 Pcs7AITC messaging

### Messaging

This block does not offer messaging.

## 17.10.5 Pcs7AITC I/Os

I/Os of Pcs7AITC

### Input parameters

| Parameter | Data type | Initial value | Description |
|---|---|---|---|
| **PV_InR** | REAL | 0 | Input value (REAL) |
| **PV_In** | WORD | 0 | Input value (WORD) |
| PV_InRSlv | REAL | 0 | Input value (REAL) of slave channel |
| PV_InSlv | WORD | 0 | Input value (WORD) of slave channel (Reserve) |
| ProImQB | BOOL | 0 | Quality bit from Process-image |
| ProImQBSlv | BOOL | 0 | Quality bit from Process-image of slave channel |
| SelQB | BOOL | 0 | 1=Select Qualitybit from Processimage |
| PV_Sel | BYTE | 1 | 0 = WORD input,<br>1 = REAL input |
| ChValueAct | BYTE | 16#00 | Channel number on the module |
| **Scale** | STRUCT<br>High: REAL<br>Low: REAL | <br>• 100.0<br>• 0.0 | Range of process value |
| **PV_InUnit** | INT | 1001 | Unit of process value |
| Feature | STRUCT<br>● Bit 0: BOOL<br>● ...<br>● Bit 31: BOOL | <br>● 0<br>● 0<br>● 0 | I/O for additional functions (Page 2554) |

| Parameter | Data type | Initial value | Description |
|---|---|---|---|
| **SimOn** | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>• 0<br>• 16#80 | 1=Simulation active |
| **SimPV_In** | STRUCT<br>Value: REAL<br>ST: BYTE | <br>• 0.0<br>• 16#80 | Simulation value |
| **SubsPV_In** | REAL | 0 | Substitution value |
| DeltaVal | REAL | 0 | Delta value (PV.value - LastValidValue) |
| **CountLim** | INT | 0 | Restart counter limit |
| **MS_Release** | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>• 0<br>• 16#80 | Maintenance release |
| **MS** | DWORD | 16#00000000 | Maintenance state |
| **MS_Ext** | DWORD | 16#00000000 | External Maintenance state |
| **TextRef** | WORD | 0 | Text reference external messages |
| FlutEn | BOOL | 0 | 1=Fluttersupress enable |
| FlutTmIn | INT | 0 | Fluttersupress time |
| FrznEn | BOOL | 0 | Enable monitoring for Frozen input value (In) |
| FrznTmIn | REAL | 0.0 | Monitoring time of Frozen input value [s] |
| DelMsgEn | BOOL | 0 | reserved |
| DelMsgTm | INT | 0 | reserved |
| SampleTime | REAL | 0.1 | Sample Time[s] |
| LowCutOff | REAL | 0.0 | Low cut off input value |
| **MrcHWC** | INT | 0 | MRC from HWC |
| **MrcReq** | BOOL | 0 | Request new measuring range center |
| **MrcVal** | INT | 0 | New value for measuring range center |
| **MrcPL** | INT | 30 | lower percentage where new MRC is requested |
| **MrcPH** | INT | 30 | upper percentage where new MRC is requested |
| **MrcTime** | INT | 12 | time (in seconds) after which a new MRC can be requested (again) |
| **MrcFrz** | INT | 8 | time (in seconds)<br>PV_Out is frozen after a change of MRC |

| Parameter | Data type | Initial value | Description |
|---|---|---|---|
| **MRC_XCHG** | ANY | | MRC information for co-ordination block |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |
| SampleTime | Sample Time [s] | REAL | 0.1 |

## Output parameters

| Parameter | Data type | Initial value | Description |
|---|---|---|---|
| **Bad** | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>0<br>16#80 | 1=Bad process value |
| **PV_Out** | STRUCT<br>Value: REAL<br>ST: BYTE | <br>0.0<br>16#80 | Process value incl. ST |
| **PV_OutUnit** | INT | 0 | Unit of process value |
| **ScaleOut** | STRUCT<br>High: REAL<br>Low: REAL | <br>100.0<br>0.0 | Range of process value |
| PV_HiAct | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>0<br>16#80 | 1=Input value high limit failure |
| PV_LoAct | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>0<br>16#80 | 1=Input value low limit failure |
| SimAct | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>0<br>16#80 | 1=Simulation active |
| ModErr | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>0<br>16#80 | 1=Higher level failure |
| FrznVal | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>0<br>16#80 | frozen input value |
| RemMonTm | REAL | 0.0 | remaining monitoring time |
| ErrorNum | INT | -1 | Parameter error |
| **OosAct** | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>0<br>16#80 | Field device out of service, maintenance in pogress |
| PV_LcoAct | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>0<br>16#80 | 1=Low cut off limit failure |

| Parameter | Data type | Initial value | Description |
|---|---|---|---|
| **MS_Req** | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>0<br>16#80 | Maintenance request |
| **MS_Dev** | DWORD | 16#00000000 | Maintenance State |
| **MrcCurr** | INT | 0 | current measuring range center |
| **MrcBusy** | BOOL | 0 | new 'measuring range center' requested |
| **MrcTiLo** | BOOL | 0 | MRC Time Lock active |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |

## In_Out parameters

| Parameter | Data type | Initial value | Description |
|---|---|---|---|
| **Mode** | DWORD | 16#00000000 | Quality and mode |
| **DataXchg** | DWORD | 16#00000000 | Data exchange |
| **DataXchg1** | DWORD | 16#00000000 | Data exchange |
| **MS_Xchg** | DWORD | 16#00000000 | Maintenance State exchange |

## Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | Parameterization |
| PV_Sel | IN | | | x |
| Mode | IN / OUT | x | | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| MS_Xchg | IN / OUT | x | | |
| MS | IN | x | | |
| MrcXchg | IN | x | | |
| SelQB | IN | | | x |
| PV_InSlv | IN | | x | |
| PV_InRSlv | IN | | x | |
| ProImQB | IN | | x | |
| ProImQBSlv | IN | | x | |
| Scale - Low | IN | | | x |
| Scale - High | IN | | | x |

| Parameter name | Type | Driver wizard – Action at the time of compilation | | |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | Parameterization |
| MrcHWC | IN | | | x |
| ChValueAct | IN | | | x |
| Feature – Bit 6 | IN | | | x |
| Feature – Bit 7 | IN | | | x |
| Feature – Bit 8 | IN | | | x |
| PV_InUnit | IN | | | x |

## See also

Parameters that cannot be connected (Page 75)

# 17.11 Pcs7AnIn - Analog input channel block

## 17.11.1 Description of Pcs7AnIn

### Object name (type + number) and family

Type + number: FB 1869

Family: Channel

### Area of application for Pcs7AnIn

The block is used for the following applications:

- Signal processing of an analog input value of S7-300/400 SM analog input modules including ET 200M Ex-io module in °C.
- Processing of the redundant input channels.

### How it works

The block cyclically processes all channel-specific signal functions of an analog input module.

It reads a raw analog value from the process image (partition) and converts it to its physical value or calculates a percentage value based on this raw value. Use the status at input parameter `Mode` to define the format of the raw value and how it is processed.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

The block is also installed automatically in the startup OB (OB100).

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameter `Mode` is interconnected to the corresponding `OMode_xx` output parameter of the MOD block.
- The in/out parameter `DataXchg` is interconnected to the corresponding `DataXchg_xx` output parameter of the MOD block.
- The `MS` parameter is interconnected to the `O_MS` output parameter of the diagnostics block.

In case of redundancy, connection with both quality bits and connection with the redundant slave channel:

- The in parameter `PV_InSlv` is interconnected with the symbol of the slave channel.
- The in parameter `ProImQB` is interconnected with the symbol of the quality bit of the master channel.
- The in parameter `ProImQBSlv` is interconnected with the symbol of the quality bit of the slave channel.

Connect the symbol generated in HW Config (symbol table) for the input channel with the `PV_In` input parameter.

In case of redundant I/O configuration, use symbol of the channel of the master module.

**Note**

If you are not using the CFC function "Generate module drivers" you must set the `Mode` in/out parameter manually. Refer to the Mode Settings for SM Modules (Page 2687) section for more on this.

If a Quality Information (QI) from the ET 200SP HA module is provided:

- the input parameter `ProImQB` is interconnected with the corresponding symbol and the input parameter `SelQB` is set to 1 (otherwise 0).

For the Pcs7AnIn block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Monitoring an analog process tag (AnalogMonitoring) (Page 2854)
- Cascade control with control loop monitoring through ConPerMon (CascadeControl) (Page 2840)
- Cascade control with PIDConR (CascadeR) (Page 2843)
- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 2827)
- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 2825)
- Override control (Page 2846)
- Override control with PIDConR (OverrideR) (Page 2848)
- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 2822)
- PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 2823)
- Ratio control with control loop monitoring through ConPerMon (RatioControl) (Page 2836)
- Ratio control with PIDConR (RatioR) (Page 2839)
- PID controller with Smith predictor (SmithPredictorControl) (Page 2830)
- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 2832)
- Step controller with direct access to the actuator and without position feedback (StepControlDirect) (Page 2831)
- Model-based predictive control (ModPreCon) (Page 2849)
- Reversible motor with controllable speed (MotorSpeedControlled) (Page 2862)
- Dosing (Dose_Lean) (Page 2856)
- Control valve (VlvAnL) (Page 2868)
- Split-range controller with control loop monitoring through ConPerMon (SplitrangeControl) (Page 2833)

### Startup characteristics

The accept value delay is started when `CountLim` ≠ 0 .

### Quality code (ET200 SP HA modules)

The quality code is read from the process image and has two states: "Good" or "Bad".

### Redundancy

The block monitors the values of the redundant signals (master and slave) with quality code and the value of the signal with "Good" quality code along with its quality code is transferred to the process.

In case both the signals deliver "Bad" quality code, the substitute value or the invalid value takes effect depending on the configuration of the Feature parameter.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

Pcs7AnIn block diagram (Page 2576)
Pcs7AnIn I/Os (Page 2572)
Pcs7AnIn messaging (Page 2572)
Pcs7AnIn error handling (Page 2571)
Pcs7AnIn functions (Page 2567)
Pcs7AnIn modes (Page 2566)

## 17.11.2 Pcs7AnIn modes

### Pcs7AnIn modes

This block does not have any modes.

### See also

Pcs7AnIn block diagram (Page 2576)
Pcs7AnIn I/Os (Page 2572)
Pcs7AnIn messaging (Page 2572)
Pcs7AnIn error handling (Page 2571)

Pcs7AnIn functions (Page 2567)
Description of Pcs7AnIn (Page 2564)

## 17.11.3 Pcs7AnIn functions

### Functions of Pcs7AnIn

The functions for this block are listed below.

### Checking the raw value

The nominal range sets the range for converting analog signals into digital values (raw values), depending on the measuring type and the range of the analog input module. The nominal range is defined in the hardware configuration and is automatically saved in the in/out parameter when the block driver `Mode` is created.

This includes an overshoot/undershoot range within which an analog signal can still be converted to a digital value. This range is defined in relation to the nominal range (roughly 18.5%). Outside this range an overflow or underflow occurs and output parameter `Bad` = 1 is set.

- Output parameter `PV_LoAct` = 1 is set if the value is outside the nominal low range.
- Output parameter `PV_HiAct` = 1 is set if the value is outside the nominal high range.

### NAMUR limit checking (only with modules of 4 to 20 mA)

In "Life Zero" monitoring, the process signal is invalid (`Bad` = 1), if the measured current is less than 3.6 mA or greater than 21 mA (defined by NAMUR).

The NAMUR limits are set as fixed defaults for limit monitoring. You can define other limits by setting input parameter `NamurOff` = 1 and setting corresponding new limits in [mA] at the `HighLimit` and `LowLimit` input parameters. If the active limits are exceeded or undershot (`HighLimit` or `LowLimit` = 1), `Bad` = 1 is set for a "Life Zero" analog signal.

**Note**

The limits that can be selected must be within the overshoot range of the module for `HighLimit` and the undershoot range of the module for `LowLimit`. Values outside the NAMUR range are also possible, if the module does not automatically limit the measured values.

### Obtaining the standard value

The standard value (a physical quantity) is obtained from the raw value using parameters `Scale` and `Mode`. Set two scale values on the structured parameter `Scale`.

- High scale value (`Scale.High`)
- Low scale value (`Scale.Low`)

If you are not using the CFC function "Generate module drivers" you must set the `Mode` in/out parameter manually. Refer to the Mode Settings for SM Modules (Page 2687) section for more on this.

The settings of the parameter `Scale` are copied to the output parameter`ScaleOut`. The output parameter can be interconnected to a corresponding input parameter of a technologic block (e.g. `PV_OpScale`).

The standard value is obtained using a linear characteristic. `Scale.Low` is the lowest physical value that the process variable can take and `Scale.High` is the highest.

If `Scale.Low` = 0 and `Scale.High` = 100 a percentage is obtained.

Special cases when obtaining the standard value using the `Scale` parameter:

- If you set `Scale.High` = `Scale.Low`, you obtain the analog input module's electrical input signal (e.g., mA) according to the `Mode` parameter setting.
- If the raw value is already a physical quantity (for example, RTD or TC), set `Scale.Low` = 0 and `Scale.High` = 1. This will cause the raw value to be output unchanged, as a physical quantity.
- When using measuring type PTC (**P**ositive **T**emperature **C**oefficient binary evaluation of resistance thermometers), the analog value contains an encoded binary signal. The `PV_Out` output provides the following information:
  - If the measured resistance is within the normal range, `PV_Out` = 0.0
  - If the measured resistance is within the prewarning range, `PV_Out` = 4.0
  - If the measured resistance is in the operating range, `PV_Out` = 1.0
    This is only true if the input parameters are `Scale.Low` = 0 and `Scale.High` = 1 . During simulation or if the substitute value is output, you can only set the input parameters `SimPV_In` and `SubsPV_In` to 0.0 or 1.0.
- Measurement type `TC_I` / `TC_E` (linearization is not taken into account) for the modules 6ES7 331-7KF0-x:
  With the measuring type "External or internal comparison of thermocouple values", the raw value is adapted to the physical variable ± 80 mV range for these modules. You have to determine the temperature using the corresponding conversion tables in the module manual. The physical equivalent in [mV] is returned by the module as a raw value. Set `Scale` to ± 80 mV.

### Holding the last value if raw value is invalid

If the block is to hold the most recent valid value when the raw value is invalid, you must activate this function at the `Feature` Bit Outputting last valid value if raw value is invalid (Page 165).

You can also influence this function via the input parameter `DeltaVal`.

- `DeltaVal` ≤ 0: the last value is retained and is not influenced
- `DeltaVal` > 0: the last or the next to last value is output

If you set the parameter `DeltaVal` > 0, the last `PV_Out`(k - 1) or next to last `PV_Out`(k - 2) valid output value is output (`PV_Out`(k) is the current value, k is the current time).

At parameter `DeltaVal` you can preset a permitted process value change (`PV_Out`) between two calls.

You have the following options:

- For invalid raw values and DeltaVal > 0:
  - If |PV_Out(k - 1) - PV_Out(k - 2)| > DeltaVal, then PV_Out = PV_Out(k - 2) (last but one valid output value is output)
  - If |PV_Out(k) - PV_Out(k - 1)| ≤ DeltaVal, then PV_Out = PV_Out(k - 1) (last valid output value is output)
- For valid raw values and DeltaVal > 0:
  - |PV_Out(k) - PV_Out(k - 1)| > DeltaVal, so for one cycle PV_Out = PV_Out(k - 1) is output, i.e. DeltaVal is used to limit the change made to the valid raw value. In addition, the signal status at the output parameter PV_Out is set to 16#60 and the output parameter is set to Bad = 0 .
    The value of DeltaVal should be selected with due care. If the value is too low, the quality code may flutter between16#80 and 16#60, regardless whether or not the raw value is OK.

### Output substitute value if raw value is invalid

If the block is to output a substitute value (SubsPV_In) when the raw value is invalid, you must activate this function at the Feature Bit Output substitute value if raw value is invalid (Page 160).

### Output of invalid value if raw value is invalid

If the block is to output an invalid value (PV_Out = PV_In), you must activate this function at the Feature Bit Output invalid raw value (Page 185).

This function is preset.

### Value application delay

After a restart, or if the output parameter Bad changes its value from 1 to 0 the signal status and the value of output parameter PV_Out are not updated until the number of cycles for delayed application of the value (input parameter CountLim) have elapsed. During the value application delay, the signal status at the output parameter is PV_Out = 16#00 and Bad = 1. The last value is retained during the value application delay.

If CountLim = 0 , the function is deactivated.

### Flutter suppression

This block provides the standard function Flutter suppression for channel blocks (Page 73)

### Signal status for PCS 7 channel blocks

The block provides the standard function Forming and outputting the signal status for PCS 7 channel blocks (Page 127).

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 5 | Activate LowCutOff (Page 166) |
| 6 | Scalable measuring range (Page 203) |
| 7 | 2 decimal points (scalable measuring range) (Page 203) |
| 8 | 3 decimal points (scalable measuring range) (Page 204) |
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 11 | Understeering / underflow for measuring range 0...10V (Page 209) |
| 12 | Selecting the Feature Bit when Namur Off = 1 (Page 212) |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |
| 28 | Output invalid raw value (Page 185) |
| 29 | Output substitute value if raw value is invalid (Page 160) |
| 30 | Outputting last valid value if raw value is invalid (Page 165) |

If `Feature.Bit5` = 1, the input parameter `LowCutOff` is compared with `PV_Out.Value`.

If `PV_Out.Value` < `LowCutOff`, then `PV_Out.Value` = `Scale.Low` and output parameter `PV_LcoAct.Value` = 1.

If `PV_Out.Value` ≥ `LowCutOff`, then `PV_LcoAct.Value` = 0.

### Sign-of-life monitoring

If an input value whose signal status is `16#80` (good) remains constant for a programmable time (monitoring time), the input value is detected as faulty and the outputs `Bad = 1` and `FrzVal = 1` are set. The signal status is set to `PV_Out.ST = 16#00`.

The monitoring time is set at the `FrznTmIn` input parameter in seconds. With `FrznTmIn = 0` or `FrznEn = 0` (default setting), the sign-of-life monitoring is deactivated, any pending errors are reset.

The input value is considered as faulty as long as it seen as constant. The monitoring time is restarted each time the input value is changed.

### Scalable measuring range

The scalable measuring range is a limited section of a measuring range supported by the module.

It allows you to increase the resolution for a configurable section.

- The "Measuring range resolution" parameter determines the resolution to 2 or 3 decimal places.
- The "Measuring range center" parameter determines the temperature over which the scalable measuring range is symmetrically spanned.

For more infromation, refer to the manual "*SIMATIC ET 200SP analog input module AI 4xRTD/TC 2-/3-/4-wire HF*".

### Delay the output "Bad"

If input `DelTiBad` > 0, on receiving an internal bad value, delay timer will start. Timer will abort if `DelTiBad` is 0. Minimum value of `DelTiBad` is 0. Only when timer expires i.e., `RemDelTiBad` = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running `Bad.ST` will be 16#60. Only after the timer expires the actual `ST` be updated. Substitute value is updated at `PV_Out`, only after delay timer expires. When timer is running last valid value will be there at `PV_Out`. When substitute value is used, `ST` of `PV_Out` will be 16#60.

### See also

Pcs7AnIn block diagram (Page 2576)

Pcs7AnIn I/Os (Page 2572)

Pcs7AnIn messaging (Page 2572)

Pcs7AnIn error handling (Page 2571)

Pcs7AnIn modes (Page 2566)

Description of Pcs7AnIn (Page 2564)

## 17.11.4 Pcs7AnIn error handling

### Error handling of Pcs7AnIn

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Channel error
- Higher-level error
- Invalid measuring range
- Frozen input value (sign-of-life monitoring)

### Channel error

At the output parameter `Bad`, channel errors are displayed with 1. Channel errors can be detected using the raw value, the NAMUR check or the sign-of-life monitoring.

`PV_LoAct` or `PV_HiAct` remain set to = 1 if a channel error occurs due to the module diagnoses "Undershoot or overshoot of the measurement range".

### Higher-level error / invalid measuring range

A higher-level error is output (output parameter `ModErr` = 1 and `Bad` = 1) if either:

- the signal status in the `High Word` of input parameter `Mode` takes the value 16#40, or
- there is an invalid measuring type in the `Low Word` of the input parameter `Mode`.

### See also

Pcs7AnIn block diagram (Page 2576)

Pcs7AnIn I/Os (Page 2572)

Pcs7AnIn messaging (Page 2572)

Pcs7AnIn functions (Page 2567)

Pcs7AnIn modes (Page 2566)

Description of Pcs7AnIn (Page 2564)

## 17.11.5 Pcs7AnIn messaging

### Messaging

This block does not offer messaging.

### See also

Pcs7AnIn block diagram (Page 2576)

Pcs7AnIn I/Os (Page 2572)

Pcs7AnIn error handling (Page 2571)

Pcs7AnIn functions (Page 2567)

Pcs7AnIn modes (Page 2566)

Description of Pcs7AnIn (Page 2564)

## 17.11.6 Pcs7AnIn I/Os

I/Os of Pcs7AnIn

## Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| CountLim | Startup counter limit | INT | 0 |
| DeltaVal | Delta value (PV_In - Last valid value) | REAL | 0.0 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 2567) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 28: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 0<br>• 0 |
| FlutEn | 1 = Flutter suppression activated | BOOL | 0 |
| FlutTmIn* | Flutter time [s] | INT | 0 |
| FrznEn | 1 = Sign-of-life monitoring activated | BOOL | 0 |
| FrznTmIn | Monitoring time in [s] | REAL | 0.0 |
| HighLimit | High limit used if NAMUR check (NamurOff = 1) is deactivated | REAL | 21.5 |
| LowLimit | Low limit used if NAMUR check (NamurOff = 1) is deactivated | REAL | 3.3 |
| LowCutOff | Input value Lowcutoff | REAL | 0.0 |
| MS | Maintenance status | DWORD | 16#00000000 |
| MS_Release | Release for maintenance (interconnected with MS_Release of the technology block) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Ext | External maintenance status. For more information, refer to the MS_Ext (Page 2463) parameter section under information on using channel blocks. | DWORD | 16#00000000 |
| NamurOff | 1= NAMUR limit verification user-defined | BOOL | 0 |
| PV_In | Process value (raw value) | WORD | 16#0000 |
| PV_InUnit* | Unit of measure for process value | INT | 1001 |
| PV_InSlv | Input value of the slave channel | WORD | 16#0000 |
| ProImQB | Quality bit from the process image | BOOL | 0 |
| ProImQBSlv | Quality bit from the process image of the slave channel | BOOL | 0 |
| SampleTime | Sampling time in [s] | REAL | 0.1 |
| Scale | Scaling of the process value as a structure | STRUCT<br>• High: REAL<br>• Low:REAL | -<br>• 100.0<br>• 0.0 |
| SelQB | 1 = Use the quality bit from the process image | BOOL | 0 |
| SimOn | 1 = Simulation on | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SimPV_In | Process value used for SimOn = 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SubsPV_In | Substitute value | REAL | 0.0 |
| TextRef | Text reference | WORD | 16#0000 |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### In/out parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| DataXchg | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |
| Mode | Value status and measuring type | DWORD | 16#00000000 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Bad | 1 = Process value is not valid | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| FrznVal | Frozen process value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see Pcs7AnIn error handling (Page 2571) | INT | -1 |
| ModErr | 1 = Device/module is faulty | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Field device is undergoing maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_HiAct | 1 = Overshoot of process value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_LcoAct | Limited to within range limits Lowcutoff | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_LoAct | 1 = Undershoot of process value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Out | Standard value (physical variable) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_OutUnit | Unit of the process value | INT | 0 |
| RemTime | Remaining monitoring time [s] | REAL | 0.0 |
| SampleTime | Sampling time [s] | REAL | 0.1 |
| ScaleOut | Scaling of the process value for display | STRUCT<br>• High: REAL<br>• Low:REAL | -<br>• 100.0<br>• 0.0 |
| SimAct | 1 = Simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |

## Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| Mode | IN / OUT | x | | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| MS_Xchg | IN / OUT | x | | |
| MS | IN | x | | |
| SelQB | IN | | | x |
| PV_InSlv | IN | | x | |
| ProImQB | IN | | x | |
| ProImQBSlv | IN | | x | |
| Scale - Low | IN | | | x |
| Scale - High | IN | | | x |

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| `Feature – Bit 6` | IN | | | x |
| `Feature – Bit 7` | IN | | | x |
| `Feature – Bit 8` | IN | | | x |
| `PV_InUnit` | IN | | | x |

**See also**

Pcs7AnIn block diagram (Page 2576)

Pcs7AnIn messaging (Page 2572)

Pcs7AnIn modes (Page 2566)

Description of Pcs7AnIn (Page 2564)

Parameters that cannot be connected (Page 75)

## 17.11.7 Pcs7AnIn block diagram

### Pcs7AnIn block diagram

A block diagram is not provided for this block.

**See also**

Pcs7AnIn I/Os (Page 2572)

Pcs7AnIn messaging (Page 2572)

Pcs7AnIn error handling (Page 2571)

Pcs7AnIn functions (Page 2567)

Pcs7AnIn modes (Page 2566)

Description of Pcs7AnIn (Page 2564)

# 17.12 Pcs7AnOu - Analog output channel block

## 17.12.1 Description of Pcs7AnOu

### Object name (type + number) and family

Type + number: FB 1870

Family: Channel

### Area of application for Pcs7AnOu

The block is used for the following applications:

- Signal processing of an analog output value from S7-300/400 SM analog output groups
- Processing of the redundant output channels

### How it works

The block outputs the process value as analog raw value for a process image (partition). Use the Mode in/out parameter to define how the raw value is to be obtained.

The current raw value is always output to the process image (partition).

Normally, the process value is written to both redundant channels.

If both redundant modules are working fine but cannot communicate with each other, the channel block Pcs7AnOu reduces the process value to 50%. And therefore, the two redundant channels use double the reduced process value 2 x 50% = 100% and not 2 x 100% = 200% which is incorrect.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameter `Mode` is interconnected to the corresponding `OMode_xx` output parameter of the MOD block.
- The in/out parameter `DataXchg` is interconnected to the corresponding `DataXchg_xx` output parameter of the MOD block.
- The `MS` parameter is interconnected to the `O_MS` output parameter of the diagnostics block.
- The `Feature.Bit0` (Set startup characteristics (Page 148)) is set when the module driver is generated in case a substitute value for this channel is parameterized in HW Config.
- Generate module driver changes the value of `StartVal` if a substitute value for this channel was parameterized in HW Config.

In case of redundancy, connection with both quality bits and connection with the redundant slave channel:

- The out parameter `PV_OutSlv` is interconnected with the symbol of the slave channel.
- The in parameter `ProImQB` is interconnected with the symbol of the quality bit of the master channel.
- The in parameter `ProImQBSlv` is interconnected with the symbol of the quality bit of the slave channel.

Connect the symbol generated in HW Config (symbol table) for the output channel with the `PV_Out` output parameter.

In case of redundant I/O configuration, use symbol of the channel of the master module.

The templates of the Advanced Process Library contain an example of an Pcs7AnOu application.

**Note**

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter `Mode` manually. Refer to the Mode Settings for SM Modules (Page 2687) chapter for more on this.

If a Quality Information (QI) from the ET 200SP HA module is provided:

- the input parameter `ProImQB` is interconnected with the corresponding symbol and the input parameter `SelQB` is set to 1 (otherwise 0).

For the Pcs7AnOu block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Cascade control with control loop monitoring through ConPerMon (CascadeControl) (Page 2840)
- Cascade control with PIDConR (CascadeR) (Page 2843)
- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 2827)
- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 2825)
- Model-based predictive control (ModPreCon) (Page 2849)
- Reversible motor with controllable speed (MotorSpeedControlled) (Page 2862)
- Override control (Page 2846)
- Override control with PIDConR (OverrideR) (Page 2848)
- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 2822)
- PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 2823)
- Ratio control with control loop monitoring through ConPerMon (RatioControl) (Page 2836)
- Ratio control with PIDConR (RatioR) (Page 2839)
- PID controller with Smith predictor (SmithPredictorControl) (Page 2830)

- Control valve (VlvAnL) (Page 2868)
- Split-range controller with control loop monitoring through ConPerMon (SplitrangeControl) (Page 2833)

### Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

### Quality code (ET200 SP HA modules)

The quality code is read from the process image and has two states: "Good" or "Bad".

### Redundancy

The block monitors the values of the redundant signals (master and slave) with quality code and delivers 100% of the application program value to both the signals:

- when one of the signals delivers "Good" quality code, or
- when both signals deliver "Good" quality code and there is no "IO Redundancy Warning" (`Mode` <> 16#x5xxxxxx) reported

In case both the signals deliver "Good" quality code and there is "IO Redundancy Warning" (`Mode` = 16#x5xxxxxx) reported then 50% of the application program value is delivered to the process.

In case both the signals deliver "Bad" quality code, 100 % of the `PV_In` value takes effect at each of the outputs `PV_Out` and `PV_OutSlv`.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

Pcs7AnOu block diagram (Page 2587)

Pcs7AnOu I/Os (Page 2584)

Pcs7AnOu messaging (Page 2583)

Pcs7AnOu error handling (Page 2582)

Pcs7AnOu modes (Page 2580)

Pcs7AnOu functions (Page 2580)

## 17.12.2 Pcs7AnOu modes

### Pcs7AnOu modes

This block does not have any modes.

### See also

Pcs7AnOu block diagram (Page 2587)
Pcs7AnOu I/Os (Page 2584)
Pcs7AnOu messaging (Page 2583)
Pcs7AnOu error handling (Page 2582)
Description of Pcs7AnOu (Page 2577)
Pcs7AnOu functions (Page 2580)

## 17.12.3 Pcs7AnOu functions

### Functions of Pcs7AnOu

The functions for this block are listed below.

### Forming an I/O value

The peripheral value `PV_Out` is formed from:

- the scale value (input parameter `Scale`)
- the process value (input parameter `PV_In`)
- the measuring type (in/out parameter `Mode`)

### Example of measuring type 4 ... 20 mA

If this measuring type is to be used, you must set the Mode parameter with 16#203 accordingly. In the measuring type, the peripheral value for 4mA is output for `PV_In` = `Scale.Low` and the peripheral value for 20 mA is output for `PV_In` = `Scale.High`.

The block writes the input parameter `Scale` directly to the output parameter `ScaleOut` and interconnects it directly to a technologic block. This can be, for example, the input parameter `MV_Opscale` of a control block.

### Limiting the process or peripheral value

The peripheral value can be limited in two different ways:

- Limited to within range limits
- Limited to scale values

**Limited to within range limits (physical limits of the module):** If you want to restrict the peripheral value (`PV_Out`), you must activate this function via the `ScaleOff` = 1 parameter.

The peripheral value is now limited to the following range limits:

- Top: 16#7EFF (32511 dec.)
- Bottom (unipolar): 0 or
- Bottom, unipolar (4 - 20 mA; 1 - 5 V): 16#E500 (-6912 dec.)
- Bottom (bipolar): 16#8100 (-32512 dec.)

If the limits are undershot or overshot `PV_HiAct` = 1 (top) or `PV_LowAct` = 1 (bottom) is displayed at the output parameters. The signal status of the `PV_ChnST` output parameter is set to 16#78 .

**Limited to scale values:** If you want to restrict the peripheral value (`PV_Out`) to the scale values, you must activate this function via the `ScaleOff` = 0 parameter. You define the high and low scale limits in the `Scale` parameter. If one of the limits is violated, the limit you have entered is output at the `PV_Out` output parameter. This is displayed at the `PV_HiAct` or `PV_LoAct` = 1 output parameter. The signal status of the `PV_ChnST` output parameter is set to 16#78.

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

### Keep last valid raw value

The block provides the standard function Keep last valid raw value if input value is bad (Page 207).

If `Feature.Bit31` is active and the input `PV_In` has bad quality (16#00 OR 16#28) the outputs `PV_Out`, `PV_OutSlv` keep its value and will not be actualized.

### Flutter suppression

This block provides the standard function Flutter suppression for channel blocks (Page 73)

### Forming the signal status for PCS 7 channel blocks

The block provides the standard function Forming and outputting the signal status for PCS 7 channel blocks (Page 127).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) section. The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 5 | Reversal of control direction for fail-open valves (Page 211) |
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |
| 30 | Outputting a de-energized value for block-external simulation (Page 159) |
| 31 | Keep last valid raw value if input value is bad (Page 207) |

### Delay the output "Bad"

If input `DelTiBad` > 0, on receiving an internal bad value, delay timer will start. Timer will abort if `DelTiBad` is 0. Minimum value of `DelTiBad` is 0. Only when timer expires i.e., `RemDelTiBad` = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running `Bad.ST` will be 16#60. Only after the timer expires the actual `ST` be updated. Substitute value is updated at `PV_Out`, only after delay timer expires. When timer is running last valid value will be there at `PV_Out`. When substitute value is used, `ST` of `PV_Out` will be 16#60.

### See also

Pcs7AnOu block diagram (Page 2587)

Pcs7AnOu I/Os (Page 2584)

Pcs7AnOu messaging (Page 2583)

Pcs7AnOu error handling (Page 2582)

Pcs7AnOu modes (Page 2580)

Description of Pcs7AnOu (Page 2577)

## 17.12.4 Pcs7AnOu error handling

### Error handling of Pcs7AnOu

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Channel error
- Higher-level error
- Invalid measuring range

### Channel error

At the output parameter `Bad`, channel errors are displayed with 1. They can be detected using the raw value or the NAMUR check or a Simatic PDM maintenance state MS=16#7xxx xxxx.

`PV_LoAct` or `PV_HiAct` remain set to = 1 if a channel error occurs due to the module diagnoses "Undershoot or overshoot of the measurement range".

### Higher-level error / invalid measuring range

A higher-level error is output (output parameter `ModErr` = 1 and `Bad` = 1) if either:

- the signal status in the `High Word` of input parameter `Mode` takes the value 16#40, or
- there is an invalid measuring type in the `Low Word` of the input parameter `Mode`.

### See also

Pcs7AnOu block diagram (Page 2587)
Pcs7AnOu I/Os (Page 2584)
Pcs7AnOu messaging (Page 2583)
Pcs7AnOu modes (Page 2580)
Description of Pcs7AnOu (Page 2577)
Pcs7AnOu functions (Page 2580)

## 17.12.5 Pcs7AnOu messaging

### Messaging

This block does not offer messaging.

### See also

Pcs7AnOu block diagram (Page 2587)
Pcs7AnOu I/Os (Page 2584)
Pcs7AnOu error handling (Page 2582)
Pcs7AnOu modes (Page 2580)

Description of Pcs7AnOu (Page 2577)

Pcs7AnOu functions (Page 2580)

## 17.12.6 Pcs7AnOu I/Os

Pcs7AnOu I/Os

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 2580) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 30: BOOL<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 0 |
| FlutEn | 1 = Flutter suppression activated | BOOL | 0 |
| FlutTmIn* | Flutter time [s] | INT | 0 |
| MS | Maintenance status | DWORD | 16#00000000 |
| MS_Release | Release for maintenance (interconnected with MS_Release of the technology block) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Ext | External maintenance status. For more information, refer to the MS_Ext (Page 2463) parameter section under information on using channel blocks. | DWORD | 16#00000000 |
| PV_In | Process value (raw value) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PV_InUnit | Unit of measure for process value | INT | 1342 |
| ProImQB | Quality bit from the process image | BOOL | 0 |
| ProImQBSlv | Quality bit from the process image of the slave channel | BOOL | 0 |
| Scale | Scaling of the process value as a structure | STRUCT<br>• High: REAL<br>• Low:REAL | -<br>• 100.0<br>• 0.0 |
| ScaleOff | 0 = Limitation to scale limits (Scale) active<br>1 = Limitation to within range limits (physical limits of the module) active | BOOL | 0 |
| SelQB | 1 = Use the quality bit from the process image | BOOL | 0 |
| SimOn | 1 = Simulation on | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SimPV_In | Process value used for SimOn = 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| StartVal | Starting value that is used on starting the block if Feature.Bit0 = 1. | REAL | 0.0 |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |
| SampleTime | Sample Time [s] | REAL | 0.1 |
| TextRef | Text reference | WORD | 16#0000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## In/out parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| DataXchg | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |
| Mode | Value status and measuring type | DWORD | 16#00000000 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Bad | 1 = Process value is not valid | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Reserved for future error numbers | INT | -1 |
| ModErr | 1 = Device/module is faulty | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |
| OosAct | 1 = Field device is undergoing maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_ChnST | Signal status of the output channel and value of PV_Out | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| PV_OutSlv | Output value of the slave channel | WORD | 16#0000 |
| PV_HiAct | 1 = Overshoot of process value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_LoAct | 1 = Undershoot of process value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Out | Process value | WORD | 16#0000 |
| PV_OutUnit | Unit of the process value | INT | 0 |
| ScaleOut | Scaling of the process value as a structure | STRUCT<br>• High: REAL<br>• Low:REAL | -<br>• 100.0<br>• 0.0 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |
| SimAct | 1 = Simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

## Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| Mode | IN / OUT | x | | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| MS_Xchg | IN / OUT | x | | |
| MS | IN | x | | |
| SelQB | IN | | | x |
| StartVal | IN | | | x |
| Feature – Bit 0 | IN | | | x |
| PV_OutSlv | OUT | | x | |
| ProImQB | IN | | x | |
| ProImQBSlv | IN | | x | |

**See also**

Pcs7AnOu block diagram (Page 2587)
Pcs7AnOu messaging (Page 2583)
Pcs7AnOu error handling (Page 2582)
Pcs7AnOu modes (Page 2580)
Description of Pcs7AnOu (Page 2577)
Parameters that cannot be connected (Page 75)

## 17.12.7 Pcs7AnOu block diagram

### Pcs7AnOu block diagram

A block diagram is not provided for this block.

**See also**

Pcs7AnOu I/Os (Page 2584)
Pcs7AnOu messaging (Page 2583)
Pcs7AnOu error handling (Page 2582)
Pcs7AnOu modes (Page 2580)
Description of Pcs7AnOu (Page 2577)
Pcs7AnOu functions (Page 2580)

# 17.13 Pcs7DiCnt - Digital input driver with count functionality

## 17.13.1 Pcs7DiCnt description

### Object name (type + number) and family

Type + number: FB 1935

Family: Channel

### Area of application

The block is used for DI count signal processing of components such as CFU or ET200SP HA.

I/O redundancy is not supported.

### How it works

The block cyclically processes all the channel specific signal functions for the count part of a DI count channel.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is installed automatically in startup OB (OB100). When the CFC function "Generate module drivers" is used after having connected `PV_InMeas` with the corresponding symbol from count address, the following operations occur automatically:

- The IN_OUT parameter `Mode` is interconnected to the corresponding `OMODE_xx` (xx = channel number) output parameter of the MOD function block.
- The IN_OUT parameter `DataXchg` and `DataXchg1` are interconnected to the corresponding `DXCHG_xx` and `DXCHG1_xx` output parameter of the MOD function block.
- The `MS` parameter is interconnected to the `O_MS` output parameter of the diagnostics block.
- The input parameter `MS_Xchg` is interconnected with `MS_XCHG_xx` of the diagnostics block.
- The input parameter `PV_InStat` is interconnected with the symbol of the corresponding input address.
- If Quality Information (QI) from ET 200SP HA module is available, the input parameter `ProImQB` is interconnected with the corresponding symbol and `SelQB` = 1 is set.

**Note**

If you are not using the CFC function "Generate module drivers" wizard. you must set the In/Out parameter `Mode` manually. Refer to the Mode Settings for SM Modules (Page 2687) chapter for more on this.

If a Quality Information (QI) from the ET 200SP HA module is provided:

- the input parameter ProImQB is interconnected with the corresponding symbol and the input parameter SelQB is set to 1 (otherwise 0).

### Startup characteristics

The block has no startup characteristics.

### Status word allocation for Status parameter

This block does not have a Status parameter.

## 17.13.2 Pcs7DiCnt functions

### Functions of Pcs7DiCnt

The functions for this block are listed below.

### Process values in Count and Frequency mode

This channel block provides two main modes called "Count" mode and "Frequency" mode. However, only one of them can be active at a time. This selection depends on the selection of the channel in HWC and is parameterized at input parameter CntMode (1=Count, 2=Frequency) by driver wizard. The provided process value also depends on this selection.

The output parameter PV_Out provides a count or a frequency value (data type REAL). The output parameter PV_OutCnt provides the count value as datatype DINT. Both are obtained by reading a process image partition.

Depending on your configuration, count channel of the module provides count values from 0 up to 4294967295. This corresponds to a data type unsigned DINT. Because STEP 7 does not provide a data type unsigned DINT, a misinterpretation of the highest bit takes place with STEP 7 data type DINT. That means, PV_OutCnt delivers values from 0...2147483647 and then goes on with -2147483648..-1. The count value is provided in parallel by output parameter PV_Out in data type REAL.

If you are not using the CFC function "Generate module drivers", you must set the In/Out parameter mode manually for further settings. Refer to the Mode Settings for SM Modules (Page 2687) section for more on this.

The function block also provides the following status information from the process image:

- PV_LdErr (1= load error of slot mechanism when setting a new count value)
- PV_StDq (status of DQ)
- PV_StGate (status of gate)
- PV_StSGate (status of software gate)

Using the input parameter SwGate, the Software Gate (SW Gate) of this channel can be set or reset.

A new count value in count mode can be predefined using the input parameter `LdVal` and can be requested with `LdReq` = 1. For this, the channel block handles the slot mechanism of the count channel.

### Holding the last value if raw value is invalid

If the block is to hold the most recent valid value when the raw value is invalid, you must activate this function at the Feature Bit Outputting last valid value if raw value is invalid (Page 165).

### Output substitute value if raw value is invalid

If the block is to output a substitute value (SubsPVCnt / SubsPCFreq) when the raw value is invalid, you must activate this function at the Feature Bit Output substitute value if raw value is invalid (Page 160)

### Output of invalid value if raw value is invalid

If the block is to output an invalid value (`PV_Cnt` = `PV_InMeas` or `PV_Freq` = `PV_InMeas`), you must activate this function at the Feature Bit Output invalid raw value (Page 185). This option is preselected.

### Signal status for PCS 7 channel blocks

The block provides the standard function Forming and outputting the signal status for PCS 7 channel blocks.

### Simulating signals

The block provides the standard function Simulating signals. In Count mode the Count value `PV_Out` / `PV_OutCnt` can be simulated, whereas in Frequency mode, the frequency value `PV_Out` can be simulated. In Count mode, the simulation value is specified with the input parameter `SimPV.Value`. If `SimPVCnt.ST` is different from 16#FF, the simulation value is specified with the input parameter `SimPVCnt.Value`.

### Configurable reactions using the Feature parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable response using the Feature I/O (Page 139) section. The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |
| 28 | Output invalid raw value (Page 185) |

| Bit | Function |
|---|---|
| 29 | Output substitute value if raw value is invalid (Page 160) |
| 30 | Outputting last valid value if raw value is invalid (Page 165) |

### Delay the output "Bad"

If input `DelTiBad` > 0, on receiving an internal bad value, delay timer will start. Timer will abort if `DelTiBad` is 0. Minimum value of `DelTiBad` is 0. Only when timer expires i.e., `RemDelTiBad` = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running `Bad.ST` will be 16#60. Only after the timer expires the actual `ST` be updated. Substitute value is updated at `PV_Out`, only after delay timer expires. When timer is running last valid value will be there at `PV_Out`. When substitute value is used, `ST` of `PV_Out` will be 16#60.

## 17.13.3 Pcs7DiCnt error handling

### Error handling of Pcs7DiCnt

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Channel error
- Higher-level error
- Invalid measuring range

### Channel error

At the output parameter `Bad`, channel errors are displayed as 1.

### Higher-level error / invalid measuring range

A higher-level error is displayed via output parameter `ModErr` = 1, if either- the signal status in the High Word of input parameter `Mode` takes the value 16#40, or there is an invalid measuring type in the Low Word of the input parameter `Mode`.

## 17.13.4 Pcs7DiCnt messaging

### Messaging

This block does not offer messaging.

## 17.13.5 Pcs7DiCnt I/Os

I/Os of Pcs7DiCnt

### Input parameters

| Parameter | Data type | Initial value | Description |
|---|---|---|---|
| **PV_InMeas** | DWORD | 16#00000000 | Counter or frequency value from module |
| PV_InStat | BYTE | 16#00 | Status information counter from module |
| ProImQB | ... | 0 | Quality Information from module |
| **CntMode** | ... | 16#00 | Count or Frequency mode (parameterized by driver wizard) |
| **LdVal** | DINT | 0 | Load value (in count mode) |
| **LdValCnt** | STRUCT<br>Value: DINT<br>ST: BYTE | <br>• L#0<br>• 16#FF | Load value (in count mode) |
| **LdReq** | BOOL | 0 | 1 = Request load value (in count mode) |
| **SwGate** | BOOL | true | Software Gate in Count mode |
| ChValueAct | BYTE | 16#00 | Channel number (parameterized by driver wizard) |
| `Feature` | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | <br>• 0<br>• 0<br>• 0 | I/O for additional functions |
| **SimOn** | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>• 0<br>• 16#80 | 1 = Simulation active |
| **SimPV** | STRUCT<br>Value: REAL<br>ST: BYTE | <br>• 0.0<br>• 16#80 | Simulation value for `PV_InMeas` |
| **SimPVCnt** | STRUCT<br>Value: DINT<br>ST: BYTE | <br>• L#0<br>• 16#FF | Simulation value for `PV_InMeas` (Count) |
| **SubsPV** | REAL | 0.0 | Substitution value for Frequency |
| **SubsPVCnt** | DINT | 0 | Substitution value for Count |
| SelQB | BOOL | 0 | 1=Select Qualitybit from Processimage |

| Parameter | Data type | Initial value | Description |
|---|---|---|---|
| **MS_Release** | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>• 0<br>• 16#80 | Maintenance release |
| **MS** | DWORD | 16#00000000 | Maintenance state |
| **MS_Ext** | DWORD | 16#00000000 | External Maintenance state |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |
| SampleTime | Sample Time [s] | REAL | 0.1 |
| **TextRef** | WORD | 16#0000 | Text reference external messages |

## Output parameters

| Parameter | Data type | Initial value | Description |
|---|---|---|---|
| **Bad** | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>• 0<br>• 16#80 | 1=Bad process value |
| **PV_Out** | STRUCT<br>Value: REAL<br>ST: BYTE | <br>• 0.0<br>• 16#80 | Count / frequency value |
| **PV_OutCnt** | STRUCT<br>Value: DINT<br>ST: BYTE | <br>• 0<br>• 16#80 | Count value |
| PV_OutLoad | DWORD | 16#00000000 | Load value (to process image) |
| PV_OutCmd | BYTE | 16#00 | Command value (to process image) |
| **PV_StDq** | BOOL | 0 | Status DQ (in count mode) |
| **PV_StGate** | BOOL | 0 | Status gate (in count mode) |
| **PV_StSGate** | BOOL | 0 | Status SW gate (in count mode) |
| **PV_LdErr** | BOOL | 0 | Load value error (in count mode) |
| SimAct | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>• 0<br>• 16#80 | 1=Simulation active |
| ModErr | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>• 0<br>• 16#80 | 1=Higher level failure |
| ErrorNum | INT | -1 | Parameter error |

| Parameter | Data type | Initial value | Description |
|---|---|---|---|
| **OosAct** | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>• 0<br>• 16#80 | Field device out of service, maintenance in progress |
| **MS_Req** | STRUCT<br>Value: BOOL<br>ST: BYTE | <br>• 0<br>• 16#80 | Maintenance request |
| `RemDelTiBad` | Remaining delay time for output Bad [s] | REAL | 0.0 |
| **MS_Dev** | DWORD | 16#00000000 | Maintenance State |

## In_Out parameters

| Parameter | Data type | Initial value | Description |
|---|---|---|---|
| **Mode** | DWORD | 16#00000000 | Quality and mode |
| **DataXchg** | DWORD | 16#00000000 | Data exchange |
| **DataXchg1** | DWORD | 16#00000000 | Data exchange |
| **MS_Xchg** | DWORD | 16#00000000 | Maintenance State exchange |

## Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| `Mode` | IN / OUT | x | | |
| `DataXchg` | IN / OUT | x | | |
| `DataXchg1` | IN / OUT | x | | |
| `MS_Xchg` | IN / OUT | x | | |
| `MS` | IN | x | | |
| `SelQB` | IN | | | x |
| `ChValueAct` | IN | | | x |
| `CntMode` | IN | | | x |
| `ProImQB` | IN | | x | |
| `PV_InStat` | IN | | x | |
| `PV_OutLoad` | OUT | | x | |
| `PV_OutCmd` | OUT | | x | |

**See also**

Parameters that cannot be connected (Page 75)

# 17.14 Pcs7DiIn - Digital input channel block

## 17.14.1 Description of Pcs7DiIn

### Object name (type + number) and family

Type + number: FB 1871

Family: Channel

### Area of application for Pcs7DiIn

The block is used for the following applications:

- Signal processing of a digital input value from S7-300/400 SM digital input groups
- Processing of the redundant input channels

### How it works

The block cyclically processes all channel-specific signal functions of a digital input module.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

The block is also installed automatically in the startup OB (OB100).

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameter `Mode` is interconnected to the corresponding `OMode_xx` output parameter of the MOD block.
- The in/out parameter `DataXchg` is interconnected to the corresponding `DataXchg_xx` output parameter of the MOD block.
- The `MS` parameter is interconnected to the `O_MS` output parameter of the diagnostics block.

In case of redundancy, connection with both quality bits and connection with the redundant slave channel:

- The in parameter `PV_InSlv` is interconnected with the symbol of the slave channel.
- The in parameter `ProImQB` is interconnected with the symbol of the quality bit of the master channel.
- The in parameter `ProImQBSlv` is interconnected with the symbol of the quality bit of the slave channel.

Connect the symbol generated in HW Config (symbol table) for the input channel with the `PV_In` input parameter.

In case of redundant I/O configuration, use symbol of the channel of the master module.

If a Quality Information (QI) from the ET 200SP HA module is provided:

- the input parameter `ProImQB` is interconnected with the corresponding symbol and the input parameter `SelQB` is set to 1 (otherwise 0).

**Note**

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter `Mode` manually. Refer to the Mode Settings for SM Modules (Page 2687) chapter for more on this.

For the Pcs7DiIn block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Monitoring eight digital process tags (Digital8Monitoring) (Page 2853)
- Monitoring of a digital process tag (DigitalMonitoring) (Page 2851)
- Reversible motor with controllable speed (MotorSpeedControlled) (Page 2862)
- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 2832)
- Step controller with direct access to the actuator and without position feedback (StepControlDirect) (Page 2831)
- Two-speed motor (Motor2Speed) (Page 2860)
- Reversing motor (MotorReversible) (Page 2861)
- Valve (Valve_Lean) (Page 2865)
- Two-way valve (Valve2Way) (Page 2866)
- Motor valve (ValveMotor) (Page 2867)
- Control valve (VlvAnL)  (Page 2868)

### Startup characteristics

The block has startup characteristics.

### Quality code (ET200 SP HA modules)

The quality code is read from the process image and has two states: “Good” or “Bad”.

### Redundancy

The block monitors the values of the redundant signals (master and slave) with quality code and:

- When one of the signals delivers “Good” quality code then the value of the signal with “Good” quality code along with its quality code is transferred to the process.
- When both the signals deliver “Good” quality code then the value of the master signal along with its quality code is transferred to the process.

In case both the signals deliver "Bad" quality code then either the substitution value or the last value depending on the selection of the user or neither when both are not selected is delivered to the process.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

Pcs7DiIn block diagram (Page 2604)
Pcs7DiIn I/Os (Page 2601)
Pcs7DiIn messaging (Page 2601)
Pcs7DiIn error handling (Page 2600)
Pcs7DiIn functions (Page 2598)
Pcs7DiIn modes (Page 2598)

## 17.14.2 Pcs7DiIn modes

### Pcs7DiIn modes

This block does not have any modes.

### See also

Pcs7DiIn block diagram (Page 2604)
Pcs7DiIn I/Os (Page 2601)
Pcs7DiIn messaging (Page 2601)
Pcs7DiIn error handling (Page 2600)
Pcs7DiIn functions (Page 2598)
Description of Pcs7DiIn (Page 2596)

## 17.14.3 Pcs7DiIn functions

### Functions of Pcs7DiIn

The functions for this block are listed below.

### Obtaining the standard value

The digital value of the process image (partition) is output at output parameter `PV_Out` with the signal status 16#80.

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter `Mode` manually for further settings. Refer to the Mode Settings for SM Modules (Page 2687) section for more on this.

### Holding the last value if raw value is invalid

If the block is to hold the most recent valid value when the raw value is invalid, you must activate this function at the `Feature` Bit Outputting last valid value if raw value is invalid (Page 165).

### Output substitute value if raw value is invalid

If the block is to output a substitute value (`SubsPV_In`) when the raw value is invalid, you must activate this function at the `Feature` Bit Output substitute value if raw value is invalid (Page 160).

### Output of invalid value if raw value is invalid

If the block is to output an invalid value (`PV_Out` = `PV_In`), you must activate this function at the `Feature` Bit Output invalid raw value (Page 185).

This function is pre-selected.

### Flutter suppression

This block provides the standard function Flutter suppression for channel blocks (Page 73)

### Signal status for PCS 7 channel blocks

The block provides the standard function Forming and outputting the signal status for PCS 7 channel blocks (Page 127).

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) section. The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |

| Bit | Function |
|---|---|
| 28 | Output invalid raw value (Page 185) |
| 29 | Output substitute value if raw value is invalid (Page 160) |
| 30 | Outputting last valid value if raw value is invalid (Page 165) |

### Delay the output "Bad"

If input `DelTiBad` > 0, on receiving an internal bad value, delay timer will start. Timer will abort if `DelTiBad` is 0. Minimum value of `DelTiBad` is 0. Only when timer expires i.e., `RemDelTiBad` = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running `Bad.ST` will be 16#60. Only after the timer expires the actual `ST` be updated. Substitute value is updated at `PV_Out`, only after delay timer expires. When timer is running last valid value will be there at `PV_Out`. When substitute value is used, `ST` of `PV_Out` will be 16#60.

### See also

Pcs7DiIn block diagram (Page 2604)

Pcs7DiIn I/Os (Page 2601)

Pcs7DiIn messaging (Page 2601)

Pcs7DiIn error handling (Page 2600)

Pcs7DiIn modes (Page 2598)

Description of Pcs7DiIn (Page 2596)

## 17.14.4 Pcs7DiIn error handling

### Error handling of Pcs7DiIn

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Channel error
- Higher-level error
- Invalid measuring range

### Channel error

At the output parameter `Bad`, channel errors are displayed with 1. They can be detected using the raw value or the NAMUR check.

### Higher-level error / invalid measuring range

A higher-level error is output (output parameter `ModErr` = 1) if either:

- the signal status in the `High Word` of input parameter `Mode` takes the value 16#40, or
- there is an invalid measuring type in the `Low Word` of the input parameter `Mode`.

### See also

Pcs7DiIn block diagram (Page 2604)
Pcs7DiIn I/Os (Page 2601)
Pcs7DiIn messaging (Page 2601)
Pcs7DiIn functions (Page 2598)
Pcs7DiIn modes (Page 2598)
Description of Pcs7DiIn (Page 2596)

## 17.14.5 Pcs7DiIn messaging

### Messaging

This block does not offer messaging.

### See also

Pcs7DiIn block diagram (Page 2604)
Pcs7DiIn I/Os (Page 2601)
Pcs7DiIn error handling (Page 2600)
Pcs7DiIn functions (Page 2598)
Pcs7DiIn modes (Page 2598)
Description of Pcs7DiIn (Page 2596)

## 17.14.6 Pcs7DiIn I/Os

I/Os of Pcs7DiIn

## Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 2598) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 28: BOOL<br>• ....<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 0<br>• 0 |
| FlutEn | 1 = Flutter suppression activated | BOOL | 0 |
| FlutTmIn* | Flutter time [s] | INT | 0 |
| MS | Maintenance status | DWORD | 16#00000000 |
| MS_Release | Release for maintenance (interconnected with MS_Release of the technology block) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Ext | External maintenance status. For more information, refer to the MS_Ext (Page 2463) parameter section under information on using channel blocks. | DWORD | 16#00000000 |
| ProImQB | Quality information from the process image | BOOL | 0 |
| PV_In | Process value (raw value) | BOOL | 0 |
| PV_InSlv | Input value of the slave channel | BOOL | 0 |
| ProImQBSlv | Quality bit from the process image of the slave channel | BOOL | 0 |
| SelQB | 1 = Use the quality bit from the process image | BOOL | 0 |
| SimOn | 1 = Simulation on | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimPV_In | Process value used for SimOn = 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SubsPV_In | Substitute value | BOOL | 0 |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |
| SampleTime | Sample Time [s] | REAL | 0.1 |
| TextRef | Text reference | WORD | 16#0000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## In/out parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| DataXchg | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |
| Mode | Value status and measuring type | DWORD | 16#00000000 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Bad | 1 = Process value is not valid | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see Pcs7DiIn error handling (Page 2600) | INT | -1 |
| ModErr | 1 = Device/module is faulty | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Field device is undergoing maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Out | Standard value (physical variable) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |
| SimAct | 1 = Simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

## Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| Mode | IN / OUT | x | | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| MS_Xchg | IN / OUT | x | | |

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| MS | IN | x | | |
| SelQB | IN | | | x |
| PV_InSlv | IN | | x | |
| ProImQB | IN | | x | |
| ProImQBSlv | IN | | x | |

**See also**

Pcs7DiIn block diagram (Page 2604)

Pcs7DiIn messaging (Page 2601)

Pcs7DiIn modes (Page 2598)

Description of Pcs7DiIn (Page 2596)

Parameters that cannot be connected (Page 75)

## 17.14.7 Pcs7DiIn block diagram

### Pcs7DiIn block diagram

A block diagram is not provided for this block.

**See also**

Pcs7DiIn I/Os (Page 2601)

Pcs7DiIn messaging (Page 2601)

Pcs7DiIn error handling (Page 2600)

Pcs7DiIn functions (Page 2598)

Pcs7DiIn modes (Page 2598)

Description of Pcs7DiIn (Page 2596)

# 17.15 Pcs7DiIT - Digital input channel block with time stamp

## 17.15.1 Description of Pcs7DiIT

### Object name (type + number) and family

Type + number: FB 1872

Family: Channel

### Area of application for Pcs7DiIT

The block is used for the following applications:

- Signal processing of a digital input value from S7-300/400 SM digital input modules with time stamp.
- Processing of the redundant input channels.

### How it works

The cyclic block processes all channel-specific signal functions of a digital input module with configured time stamp.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

The block is also installed automatically in the startup OB (OB100).

Connect the symbol generated in HW Config (symbol table) for the input channel with the `PV_In` input parameter.

In case of redundant I/O configuration, use symbol of the channel of the master module.

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameter `Mode` is interconnected to the corresponding `OMode_xx` output parameter of the MOD block.
- The in/out parameter `DataXchg` is interconnected to the corresponding `DataXchg_xx` output parameter of the MOD block.
- The `MS` parameter is interconnected to the `O_MS` output parameter of the diagnostics block.
- If the process image (partition) also contains the value status (status bit) of the digital input channel, the corresponding symbol is connected with input `ProImQB` and the input `SelQB` = 1 is set.

- The parameter `TS_In` is interconnected to the output parameter `TS_XX` of the IMDRV_TS block or the parameter `TS_IN_PN` is interconnected to the output parameter `TS_XX` of the IM_TS_PN block.
- The in/out parameter `TS_C` is interconnected to the corresponding `TS_C_xx` output parameter of the IMDRV_TS or IM_TS_PN block.

If a Quality Information (QI) from the ET 200SP HA module is provided:

- the input parameter `ProImQB` is interconnected with the corresponding symbol and the input parameter `SelQB` is set to 1 (otherwise 0).

**Note**

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter `Mode` manually. Refer to the Mode Settings for SM Modules (Page 2687) chapter for more on this.

In case of redundancy, connection with both quality bits and connection with the redundant slave channel:

- The in parameter `PV_InSlv` is interconnected with the symbol of the slave channel.
- The in parameter `ProImQB` is interconnected with the symbol of the quality bit of the master channel.
- The in parameter `ProImQBSlv` is interconnected with the symbol of the quality bit of the slave channel.

If the time stamp for the input channel is deactivated, the aforementioned connections to the block IMDRV_TS or IM_TS_PN are not used. The block can be used as a simple channel block. When the charts are compiled, a warning is issued as the module driver is generated.

Refer to the section on configuring the channel blocks for information on configuration.

### Startup characteristics

The block has startup characteristics.

### Quality code (ET200 SP HA modules)

The quality code is read from the process image and has two states: “Good” or “Bad”.

### Redundancy

The block monitors the values of the redundant signals (master and slave) with quality code and:

- When one of the signals delivers “Good” quality code then the value of the signal with “Good” quality code along with its quality code is transferred to the process.
- When both the signals deliver “Good” quality code then the value of the master signal along with its quality code is transferred to the process.

In case both the signals deliver "Bad" quality code then either the substitution value or the last value depending on the selection of the user or neither when both are not selected is delivered to the process.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

Pcs7DiIT block diagram (Page 2614)

Pcs7DiIT I/Os (Page 2611)

Pcs7DiIT messaging (Page 2610)

Pcs7DiIT error handling (Page 2610)

Pcs7DiIT functions (Page 2607)

Pcs7DiIT modes (Page 2607)

## 17.15.2 Pcs7DiIT modes

### Pcs7DiIT modes

This block does not have any modes.

### See also

Pcs7DiIT block diagram (Page 2614)

Pcs7DiIT I/Os (Page 2611)

Pcs7DiIT messaging (Page 2610)

Pcs7DiIT error handling (Page 2610)

Pcs7DiIT functions (Page 2607)

Description of Pcs7DiIT (Page 2605)

## 17.15.3 Pcs7DiIT functions

### Functions of Pcs7DiIT

The functions for this block are listed below.

### Obtaining the standard value

The digital value of the process image (partition) is output at output parameter `PV_Out` with the signal status 16#80 .

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter `Mode` manually for further settings. Refer to the section Mode Settings for SM Modules (Page 2687).

### Holding the last value if raw value is invalid

If the block is to hold the most recent valid value when the raw value is invalid, you must activate this function at the `Feature` Bit Outputting last valid value if raw value is invalid (Page 165).

### Output substitute value if raw value is invalid

If the block is to output a substitute value (`SubsPV_In`) when the raw value is invalid, you must activate this function at the `Feature` Bit Output substitute value if raw value is invalid (Page 160).

### Output of invalid value if raw value is invalid

If the block is to output an invalid value (`PV_Out` = `PV_In`), you must activate this function at the `Feature` Bit Output invalid raw value (Page 185).

This function is pre-selected.

### Flutter suppression

This block provides the standard function Flutter suppression for channel blocks (Page 73)

### Signal status for PCS 7 channel blocks

The block provides the standard function Forming and outputting the signal status for PCS 7 channel blocks (Page 127).

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

### Time stamp

The block provides the standard function Time stamp (Page 231).

- Connect the symbol generated in HW Config (symbol table) for the input channel with the `PV_In` input parameter. (...)
- The `TS_In` parameter is interconnected to the `TS_XX` output parameter of the `IMDRV_TS` or `IM_TS_PN` block.
- The in/out parameter `TS_C` is interconnected to the corresponding `TS_C_xx` output parameter of the `IMDRV_TS` or `IM_TS_PN` block.

- If the time stamp of the I/O is to be used with `TimeStampOn` = 1(I/O device time) set for an `EventTS` instance, it must be ensured that `Feature.Bit5` = 0 (I/O device time) for all connected `Pcs7DiIT` channel blocks.
- If the CPU-Time is to be used with `TimeStampOn` = 0(CPU-Time) set for an `EventTS` instance, it must be ensured that `Feature.Bit5` = 1 (CPU-Time) for all connected `Pcs7DiIT` channel blocks. In addition the time-stamping for all such channels has to be deactivated in HW Configuration to avoid IM buffer overflow or handshaking problems caused by such unused but activated time-stamped signals.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the section Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 5 | Timestamp calculation (Page 210) |
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |
| 28 | Output invalid raw value (Page 185) |
| 29 | Output substitute value if raw value is invalid (Page 160) |
| 30 | Outputting last valid value if raw value is invalid (Page 165) |

### Delay the output "Bad"

If input `DelTiBad` > 0, on receiving an internal bad value, delay timer will start. Timer will abort if `DelTiBad` is 0. Minimum value of `DelTiBad` is 0. Only when timer expires i.e., `RemDelTiBad` = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running `Bad.ST` will be 16#60. Only after the timer expires the actual `ST` be updated. Substitute value is updated at `PV_Out`, only after delay timer expires. When timer is running last valid value will be there at `PV_Out`. When substitute value is used, `ST` of `PV_Out` will be 16#60.

### See also

Pcs7DiIT block diagram (Page 2614)

Pcs7DiIT I/Os (Page 2611)

Pcs7DiIT messaging (Page 2610)

Pcs7DiIT error handling (Page 2610)

Pcs7DiIT modes (Page 2607)

Description of Pcs7DiIT (Page 2605)

Evaluation of the signal status of the interlock signals (Page 153)

## 17.15.4 Pcs7DiIT error handling

### Error handling of Pcs7DiIT

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Channel error
- Higher-level error
- Invalid measuring range

### Channel error

At the output parameter `Bad`, channel errors are displayed with 1. The channel error is generated from the signal status `PV_ST`.

### Higher-level error / invalid measuring range

A higher-level error is displayed at the output parameters `ModErr` and `Bad` with 1 if the signal status at `High Word` of the input parameter `Mode` accepts the value 16#40.

A higher-level error is also present when an incorrect measuring type is entered in `Low Word` of the input parameter `Mode`.

### See also

Pcs7DiIT block diagram (Page 2614)

Pcs7DiIT I/Os (Page 2611)

Pcs7DiIT messaging (Page 2610)

Pcs7DiIT functions (Page 2607)

Pcs7DiIT modes (Page 2607)

Description of Pcs7DiIT (Page 2605)

## 17.15.5 Pcs7DiIT messaging

### Messaging

This block does not offer messaging.

### See also

Pcs7DiIT block diagram (Page 2614)

Pcs7DiIT I/Os (Page 2611)

Pcs7DiIT error handling (Page 2610)

Pcs7DiIT functions (Page 2607)

Pcs7DiIT modes (Page 2607)

Description of Pcs7DiIT (Page 2605)

## 17.15.6 Pcs7DiIT I/Os

**I/Os of Pcs7DiIT**

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 2607) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 28: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 0<br>• 0 |
| FlutEn | 1 = Flutter suppression activated | BOOL | 0 |
| FlutTmIn* | Flutter time [s] | INT | 0 |
| MS | Maintenance status | DWORD | 16#00000000 |
| MS_Ext | External maintenance status. For more information, refer to the MS_Ext (Page 2463) parameter section under information on using channel blocks. | DWORD | 16#00000000 |
| MS_Release | Release for maintenance (interconnected with MS_Release of the technology block) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_In | Process value (raw value) | BOOL | 0 |
| PV_InSlv | Input value of the slave channel | BOOL | 0 |
| ProImQB | Quality bit from the process image | BOOL | 0 |
| ProImQBSlv | Quality bit from the process image of the slave channel | BOOL | 0 |
| SimOn | 1 = Simulation on | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| SimPV_In | Process value used for SimOn = 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SelQB | 1 = Use the quality bit from the process image | BOOL | 0 |
| SubsPV_In | Substitute value | BOOL | 0 |
| TextRef | Text reference | WORD | 16#0000 |
| TS_In | Time stamp of IMDRV_TS;<br>The time is in ISP format | STRUCT<br>• MsgSig: BOOL<br>• TriInf: BOOL<br>• HdSh: BOOL<br>• ST: BYTE<br>• TS0: DWORD<br>• TS1: DWORD | -<br>• 0<br>• 0<br>• 0<br>• 16#80<br>• 16#00000000<br>• 16#00000000 |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |
| SampleTime | Sample Time [s] | REAL | 0.1 |
| TS_In_PN | Time stamp from IM_TS_PN<br>The time is in ISP format | STRUCT<br>• MsgSig: BOOL<br>• TriInf: BOOL<br>• HdSh: BOOL<br>• ST: BYTE<br>• TS0: DWORD<br>• TS1: DWORD<br>• TS2: DWORD | -<br>• 0<br>• 0<br>• 0<br>• 16#80<br>• 16#00000000<br>• 16#00000000<br>• 16#00000000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### In/out parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| DataXchg | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |
| TS_C | Time-stamp communication | BYTE | 16#00 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Bad | 1 = Process value is not valid | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Reserved for future error numbers | INT | -1 |
| ModErr | 1 = Device/module is faulty | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Field device is undergoing maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Out | 1 = Standard value (physical variable) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1 = Simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |
| TS_Out | Time stamp | STRUCT<br>• MsgSig: BOOL<br>• TriInf: BOOL<br>• HdSh: BOOL<br>• ST: BYTE<br>• TS0: DWORD<br>• TS1: DWORD<br>• Link: BOOL | -<br>• 0<br>• 0<br>• 0<br>• 16#80<br>• 16#00000000<br>• 16#00000000<br>• 0 |

### Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| Mode | IN / OUT | x | | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| MS_Xchg | IN / OUT | x | | |
| MS | IN | x | | |
| SelQB | IN | | | x |
| PV_InSlv | IN | | x | |
| ProImQB | IN | | x | |
| ProImQBSlv | IN | | x | |
| TS_In / TS_In_PN | IN | x | | |
| TS_C | IN / OUT | x | | |

### See also

Pcs7DiIT block diagram (Page 2614)

Pcs7DiIT messaging (Page 2610)

Pcs7DiIT error handling (Page 2610)

Pcs7DiIT modes (Page 2607)

Description of Pcs7DiIT (Page 2605)

Parameters that cannot be connected (Page 75)

## 17.15.7 Pcs7DiIT block diagram

### Pcs7DiIT block diagram

A block diagram is not provided for this block.

### See also

Pcs7DiIT I/Os (Page 2611)

Pcs7DiIT messaging (Page 2610)

Pcs7DiIT error handling (Page 2610)

Pcs7DiIT functions (Page 2607)
Pcs7DiIT modes (Page 2607)
Description of Pcs7DiIT (Page 2605)

# 17.16 Pcs7DiOu - Digital output channel block

## 17.16.1 Description of Pcs7DiOu

### Object name (type + number) and family

Type + number: FB 1873

Family: Channel

### Area of application for Pcs7DiOu

The block is used for the following applications:

- Signal processing of a digital output value from S7-300/400 SM digital output groups.
- Processing of the redundant output channels.

### How it works

The cyclic block processes all channel-specific signal functions of a digital output module.

Normally, the process value is written to both redundant channels.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

The block is also installed automatically in the startup OB (OB100).

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameter `Mode` is interconnected to the corresponding `OMode_xx` output parameter of the MOD block.
- The in/out parameter `DataXchg` is interconnected to the corresponding `DataXchg_xx` output parameter of the MOD block.
- The `MS` parameter is interconnected to the `O_MS` output parameter of the diagnostics block.
- The `Feature.Bit0` (Set startup characteristics (Page 148)) is set when the module driver is generated in case a substitute value for this channel is parameterized in HW Config.
- Generate module driver changes the value of `StartVal` if a substitute value for this channel was parameterized in HW Config.

In case of redundancy, connection with both quality bits and connection with the redundant slave channel:

- The out parameter `PV_OutSlv` is interconnected with the symbol of the slave channel.
- The in parameter `ProImQB` is interconnected with the symbol of the quality bit of the master channel.
- The in parameter `ProImQBSlv` is interconnected with the symbol of the quality bit of the slave channel.

Connect the symbol generated in HW Config (symbol table) for the output channel with the `PV_Out` output parameter.

In case of redundant I/O configuration, use symbol of the channel of the master module.

The templates of the Advanced Process Library contain an example of an Pcs7DiOu application.

If a Quality Information (QI) from the ET 200SP HA module is provided:

- the input parameter `ProImQB` is interconnected with the corresponding symbol and the input parameter `SelQB` is set to 1 (otherwise 0).

**Note**

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter `Mode` manually. Refer to the Mode Settings for SM Modules (Page 2687) chapter for more on this.

For the Pcs7DiOu block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Dosing (Dose_Lean) (Page 2856)
- Two-speed motor (Motor2Speed) (Page 2860)
- Reversing motor (MotorReversible) (Page 2861)
- Reversible motor with controllable speed (MotorSpeedControlled) (Page 2862)
- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 2832)
- Step controller with direct access to the actuator and without position feedback (StepControlDirect) (Page 2831)
- Valve (Valve_Lean) (Page 2865)
- Two-way valve (Valve2Way) (Page 2866)
- Motor valve (ValveMotor) (Page 2867)

## Startup characteristics

Use the `Feature` Bit Set startup characteristics (Page 148) to define the startup characteristics of this block.

### Quality code (ET200 SP HA modules)

The quality code is read from the process image and has two states: “Good” or “Bad”.

### Redundancy

The block monitors the values of the redundant signals (master and slave) with quality code and the application program value is deliverd to both the signals when one or both of the signals deliver(s) “Good” quality code along with the quality code is transferred to the process.

In case both the signals deliver “Bad” quality code, the application program value is delivered to the process with “Bad” quality code.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

Pcs7DiOu block diagram (Page 2624)
Pcs7DiOu I/Os (Page 2621)
Pcs7DiOu messaging (Page 2621)
Pcs7DiOu error handling (Page 2620)
Pcs7DiOu functions (Page 2619)
Pcs7DiOu modes (Page 2618)

## 17.16.2 Pcs7DiOu modes

### Pcs7DiOu modes

The block does not have any operating modes.

### See also

Pcs7DiOu block diagram (Page 2624)
Pcs7DiOu I/Os (Page 2621)
Pcs7DiOu messaging (Page 2621)
Pcs7DiOu error handling (Page 2620)
Pcs7DiOu functions (Page 2619)
Description of Pcs7DiOu (Page 2616)

## 17.16.3 Pcs7DiOu functions

### Functions of Pcs7DiOu

The functions for this block are listed below.

### Forming a peripheral value

The digital value is written to the process image (partition). The signal status of the process value (`PV_ChnST`) is set to "good" (16#80).

### Simulating signals

The block provides the standard function Simulating signals (Page 64).

### Keep last valid raw value

The block provides the standard function Keep last valid raw value if input value is bad (Page 207).

If `Feature.Bit31` is active and the input `PV_In` has bad quality (16#00 OR 16#28) the outputs `PV_Out`, `PV_OutSlv` keep its value and will not be actualized.

### Flutter suppression

This block provides the standard function Flutter suppression for channel blocks (Page 73)

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in section Configurable functions using the Feature I/O (Page 139). The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 0 | Set startup characteristics (Page 148) |
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |
| 30 | Outputting a de-energized value for block-external simulation (Page 159) |
| 31 | Keep last valid raw value if input value is bad (Page 207) |

### Delay the output "Bad"

If input `DelTiBad` > 0, on receiving an internal bad value, delay timer will start. Timer will abort if `DelTiBad` is 0. Minimum value of `DelTiBad` is 0. Only when timer expires i.e., `RemDelTiBad` = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running `Bad.ST` will be 16#60. Only after the timer expires the actual `ST` be updated. Substitute value is updated at `PV_Out`, only after delay timer expires. When timer is running last valid value will be there at `PV_Out`. When substitute value is used, `ST` of `PV_Out` will be 16#60.

### See also

Pcs7DiOu block diagram (Page 2624)

Pcs7DiOu I/Os (Page 2621)

Pcs7DiOu messaging (Page 2621)

Pcs7DiOu error handling (Page 2620)

Pcs7DiOu modes (Page 2618)

Description of Pcs7DiOu (Page 2616)

## 17.16.4 Pcs7DiOu error handling

### Error handling of Pcs7DiOu

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Channel error
- Higher-level error
- Invalid measuring range

### Channel error

At the output parameter `Bad`, channel errors are displayed with 1. The channel error is generated from the signal status `PV_ST`.

### Higher-level error / invalid measuring range

A higher-level error is output (output parameter `ModErr` = 1) if either:

- the signal status in the `High Word` of input parameter `Mode` takes the value 16#40, or
- there is an invalid measuring type in the `Low Word` of the input parameter `Mode`.

### See also

Pcs7DiOu block diagram (Page 2624)
Pcs7DiOu I/Os (Page 2621)
Pcs7DiOu messaging (Page 2621)
Pcs7DiOu functions (Page 2619)
Pcs7DiOu modes (Page 2618)
Description of Pcs7DiOu (Page 2616)

## 17.16.5 Pcs7DiOu messaging

### Messaging

This block does not offer messaging.

### See also

Pcs7DiOu block diagram (Page 2624)
Pcs7DiOu I/Os (Page 2621)
Pcs7DiOu error handling (Page 2620)
Pcs7DiOu functions (Page 2619)
Pcs7DiOu modes (Page 2618)
Description of Pcs7DiOu (Page 2616)

## 17.16.6 Pcs7DiOu I/Os

I/Os of Pcs7DiOu

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions (Page 2619) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 30: BOOL<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 0 |
| FlutEn | 1 = Flutter suppression activated | BOOL | 0 |
| FlutTmIn* | Flutter time [s] | INT | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| MS | Maintenance status | DWORD | 16#00000000 |
| MS_Release | Release for maintenance (interconnected with MS_Release of the technology block) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Ext | External maintenance status. For more information, refer to the MS_Ext (Page 2463) parameter section under information on using channel blocks. | DWORD | 16#00000000 |
| PV_In | Process value (raw value) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ProImQB | Quality bit from the process image | BOOL | 0 |
| ProImQBSlv | Quality bit from the process image of the slave channel | BOOL | 0 |
| SelQB | 1 = Use the quality bit from the process image | BOOL | 0 |
| SimOn | 1 = Simulation on | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimPV_In | Process value used for SimOn = 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| StartVal | Starting value that is used on starting the block if Feature.Bit0 = 1. | BOOL | 0 |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |
| SampleTime | Sample Time [s] | REAL | 0.1 |
| TextRef | Text reference | WORD | 16#0000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## In/out parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| DataXchg | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |
| Mode | Value status and measuring type | DWORD | 16#00000000 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Bad | 1 = Process value is not valid | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number. For error numbers that can be output by this block, see Pcs7DiOu error handling (Page 2620) | INT | -1 |
| ModErr | 1 = Device/module is faulty | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | 1 = Field device is undergoing maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_ChnST | Signal status of the output channel and value of PV_Out | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Out | Process value | BOOL | 0 |
| PV_OutSlv | Output value of the slave channel | BOOL | 0 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |
| SimAct | 1 = Simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

## Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| Mode | IN / OUT | x | | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| S_Xchg | IN / OUT | x | | |
| MS | IN | x | | |
| SelQB | IN | | | x |

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| StartVal | IN | | | x |
| Feature – Bit 0 | IN | | | x |
| PV_OutSlv | OUT | | x | |
| ProImQB | IN | | x | |
| ProImQBSlv | IN | | x | |

**See also**

Pcs7DiOu block diagram (Page 2624)

Pcs7DiOu messaging (Page 2621)

Pcs7DiOu modes (Page 2618)

Description of Pcs7DiOu (Page 2616)

Parameters that cannot be connected (Page 75)

## 17.16.7 Pcs7DiOu block diagram

### Pcs7DiOu block diagram

A block diagram is not provided for this block.

**See also**

Pcs7DiOu I/Os (Page 2621)

Pcs7DiOu messaging (Page 2621)

Pcs7DiOu error handling (Page 2620)

Pcs7DiOu functions (Page 2619)

Pcs7DiOu modes (Page 2618)

Description of Pcs7DiOu (Page 2616)

# 17.17 Pcs7Cnt1 - Controlling and reading FM 350 modules

## 17.17.1 Description of Pcs7Cnt1

### Object name (type + number)

Type + number: FB 1833

Family: Channel

### Area of application

The Pcs7Cnt1 block is used for controlling and reading count or measured values of an FM 350-1 or FM 350-2 module.

### How it works

"FM 350" refers to the FM 350-1 and FM 350-2 modules in the following.

- The block only communicates through the process image for the FM 350-1. The data are written and read continuously.
- The control and status information and selected count and measured values are contained in the process image for the FM 350-2. The remaining count and measured values can be read via data records.
  In HW Config (User_Type1 and User_Type2) you define how the count or measured values will be saved in the process image. The `LoadPv1` and `CmpVx` parameters are loaded from the FM_CNT block to the FM 350-2 using data records. The writing of the parameters is first triggered in the subsequent cycle of the FM_CNT block.

If an FM 350-2 module is being used, the block writes the `LoadPV1` (load count value immediately) or `CmpVx` (comparison value) parameters to the module (x = channel number) via data records. If the parameter `LoadDir` = 1 is set in the block, it writes `LoadPV1`. If `LoadPre` = 1 is set, it writes in preparation `LoadPV1`. The `CmpVx` parameter is written after every change.

The `Mode` input indicates in what format the count and/or measured value is available in the process image. If the high word of the input parameter `Mode` = 16#40xxxx (value status = higher-level error, `ModErr`= 1), the count or measured value is treated as invalid.

The measured values are written to the corresponding outputs, `PV1`, `PV1_Li` as well as `PV2` and `PV2_Li`; a differential value between the old value and new value is formed for the `PV1` via the last cycle and sent at the `PV1CycLi` output. If the read values are in order, the status of the outputs is set to 16#80.

The units for PV1_UnitLi, ScalePV1_Li, PV2_UnitLi and ScalePV2_Li are set with the extension "_IN" via the inputs of the same name.

**Note**

Status CmpVal0(comparator 1), CmpVal1 (comparator 2), ZeroSt (zero crossing), OFlow (overflow) and UFlow (underflow) are automatically acknowledged. They are active for at least one cycle.

The measured value is output as a numeric value by the FM 350. Additional information on this is available in the manual for the module.

The block also supplies the counted pulses of PV1(PV1_Li) at the PV1CycLi output with each block call.

The LoadPV1 input parameter is the relevant value which is passed to the module. If, however, LoadPV1_Li is interconnected or the status of LoadPV1_Li is 16#80 , the value is written to LoadPV1 and applied.

**Note**

The display in PCS 7 is limited by the data type DINT. However, the settings in the module are permitted unsigned up to 32 bits.

## Configuration

When the CFC function "**Generate module drivers**" is used, the following occurs automatically:

- The Laddr and Channel inputs are configured.
- The Mode input is interconnected with the OMODEx output of the FM_CNT block.
- The FM_DATA structure is interconnected with the structure of the same name of the FM_CNT block.
- The DataXchg input is interconnected to the relevant DXCHG_00 output of the MOD_D1 block.
- The MS input is interconnected to the relevant O_MS output of the MOD_D1 block.

The block is installed in the CFC editor in a cyclic interrupt OB (OB30 to OB38) .

## Startup characteristics

The block does not have any startup characteristics.

## Status word allocation

This block does not have the Status parameter.

## See also

Pcs7Cnt1 I/Os (Page 2630)

## 17.17.2 Pcs7Cnt1 modes

### Pcs7Cnt1 operating modes

This block does not have any modes.

## 17.17.3 Pcs7Cnt1 functions

### Functions of Pcs7Cnt1

The functions for this block are listed below.

### Addressing

**1**. Create icons for the required count or measured values in the icon table, in accordance with the base address of the FM 350 module. Please note the following:

- FM 350-1: Count or measured value is always in the process image:
  - Select "ED base address" of the module (e.g. ED512) as the address.
- FM 350-2: Count or measured value of the desired channel is in the process image:
  - In "HW Config FM 350-2 configure counter" you can specify where count or measured values will be stored in the process image. Depending on the configuration of User_Type1 or User_Type2, you must select EW for WORD or ED for DWORD. The address is calculated according to the following table:

| Count or measured value is defined as: | Measured or count value is in User_Type1: | Measured or count value is in User_Type2: |
|---|---|---|
| DWORD or LOW WORD | FM 350-2 base address + 8 bytes | FM 350-2 base address + 12 bytes |
| HIGH WORD | FM 350-2 base address + 10 bytes | FM 350-2 base address + 14 bytes |
| **Example:**<br>The desired count value of channel 2 is in User_Type2 in the high word. The address is calculated for a base address of 512 as: Address = EW 526. | | |

- FM 350-2: Count or measured value of the desired channel is not in the process image:
  - Select as address: Input word "Base address of the module + channel number" interconnected (e.g. base address = 512, channel number = 5; EW517).

**2**. Connect the input `Connect` in the CFC chart with the previously created icon via "Interconnection to address...".

Count and measured values that are not in the process image of the FM 350-2 are read out of the module cyclically as a data record, if the inputs `PV1_EN` or `PV2_EN` = 1. Both inputs should be set to 0 for performance reasons, if count or measured value are not needed for the channel in

the user program. This prevents count or measured values from being read via data records, if they are not in the process image.

**Note**

Even if the PV1_EN or PV2_EN inputs are not set, a read can be performed via data records if the PV1_EN or PV2_EN are set inputs are set at a different instance of Pcs7Cnt1 (other channel) of the associated FM350-2.

## Simulation

The block provides the standard function Simulating signals (Page 64)

With SimOn = 1 , the simulation values SimPV1 and SimPV2 are written to the outputs PV1, PV1_Li as well as PV2 and PV2_Li . The status of the outputs is set to 16#60 and Bad = 1 in the process.

Simulation takes highest priority.

If the block is in simulation state, SimAct = 1 is set.

## Flutter suppression

This block provides the standard function Flutter suppression for channel blocks (Page 73).

## Configurable reactions using the Feature parameter

You can find an overview of all reactions provided by the Feature parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |
| 28 | Output invalid raw value (Page 185) |
| 29 | Output substitute value if raw value is invalid (Page 160) |
| 30 | Outputting last valid value if raw value is invalid (Page 165) |

## Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for channel blocks for field devices (Page 127).

| Value | Meaning |
|---|---|
| 16#80 | Valid value |
| 16#60 | Manipulated value (for example, substitute value, simulation, last valid value) |
| 16#00 | Invalid value |

The worst signal status ST_Worst for the block is formed from the following parameters:

- LoadDir.ST
- LoadPre.ST
- LoadPV1_Li.ST
- CntRun.ST
- CntDir.ST
- PV1_Li.ST
- PV2_Li.ST

### Delay the output "Bad"

If input DelTiBad > 0, on receiving an internal bad value, delay timer will start. Timer will abort if DelTiBad is 0. Minimum value of DelTiBad is 0. Only when timer expires i.e., RemDelTiBad = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running Bad.ST will be 16#60. Only after the timer expires the actual ST be updated. Substitute value is updated at PV_Out, only after delay timer expires. When timer is running last valid value will be there at PV_Out. When substitute value is used, ST of PV_Out will be 16#60.

## 17.17.4 Pcs7Cnt1 error handling

### Error handling of Pcs7Cnt1

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Higher-level error

### Overview of error numbers

The ErrorNum I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. |
| 21 | Pulse counter overflow |

### Higher-level error

A higher-level error is displayed at the output parameters `ModErr` and `Bad` with 1 if the signal status in the high word of the input parameter `Mode` assumes the value 16#40.

In addition, at the output parameter `PV_Li` of the signal status, either 16#00 (in the event of an error) or 16#60 (for manipulated value (for example, substitute value, simulation, last valid value) is output.

## 17.17.5 Pcs7Cnt1 messaging

### Messaging

This block does not offer messaging.

## 17.17.6 Pcs7Cnt1 I/Os

I/Os of Pcs7Cnt1

### Input parameters

| Parameter | Meaning | Type | Default |
|---|---|---|---|
| Channel | Channel FM 350 | INT | 0 |
| **CmpV0** | New comparison value 0 | DINT | 0 |
| CmpV1 | New comparison value 1 | DINT | 0 |
| CmpV2 | New comparison value 2 | DINT | 0 |
| CmpV3 | New comparison value 3 | DINT | 0 |
| **Connect** | Connection value (WORD or DWORD) | ANY | |
| **CtrlDO0** | 1 = Enable digital output DO | BOOL | 1 |
| CtrlDO1 | 1 = Enable digital output DO1 (FM 350-1 or FM 350-2 only, dosing mode) | BOOL | 1 |
| CtrlDO2 | 1 = Enable digital output DO2 (FM 350-2 only, dosing mode) | BOOL | 1 |
| CtrlDO3 | 1 = Enable digital output DO3 (FM 350-2 only, dosing mode) | BOOL | 1 |
| EnSetDn | 1 = enable for setting in reverse direction | BOOL | 1 |
| EnSetUp | 1 = Enable for setting in forward direction | BOOL | 1 |
| `Feature` | I/O for additional functions (Page 2627) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 28: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 0<br>• 0 |
| FlutEn | 1 = Flutter suppression active | BOOL | 0 |

| Parameter | Meaning | Type | Default |
|---|---|---|---|
| FlutTmIn* | Flutter suppression time | INT | 0 |
| Laddr | Logical address FM 350 | INT | 0 |
| LoadPv1* | Load counter | DINT | 0 |
| **LoadPV1_Li** | Load counter structured | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| **LoadDir*** | 1 = Load counter directly | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **LoadPre*** | 1 = Load prepared counter | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **MS** | Maintenance status | DWORD | 16#00000000 |
| **MS_Release** | Release for maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **MS_Ext** | External maintenance status. For more information, refer to the `MS_Ext` (Page 2463) parameter section under information on using channel blocks. | DWORD | 16#00000000 |
| **PV1En** | 1 = Counted value used | BOOL | 1 |
| **PV2En** | 1 = Measured value used | BOOL | 1 |
| **PV1_Unit** | Unit of the count value | INT | 0 |
| **PV2_Unit** | Unit of the measured value | INT | 0 |
| **RstOpErr** | 1 = Reset operator error | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **RstSync*** | 1 = Reset synchronization | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **ScalePV1** | Range of the current count value | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| **ScalePV2** | Range of current measured value | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| **SetDO0** | 1 = Open DO0 | BOOL | 1 |
| SetDO1 | 1 = Open DO1 (FM 350-1 or FM 350-2 only, dosing mode) | BOOL | 1 |
| SetDO2 | 1 = Open DO2 (FM 350-2 only, dosing mode) | BOOL | 1 |
| SetDO3 | 1 = Open DO3 (FM 350-2 only, dosing mode) | BOOL | 1 |

| Parameter | Meaning | Type | Default |
|---|---|---|---|
| **SimPV1** | Simulation count value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| **SimPv2** | Simulation measured value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| **SimON** | 1 = simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Stopgate | 1 = General GATE stop | BOOL | 0 |
| **StopGate** | 1 = Stop gate | BOOL | 1 |
| **SubsPV1** | Count substitute value | DINT | 0 |
| **SubsPV2** | Measured substitute value | DINT | 0 |
| SubsOn | 1 = substitute value active | BOOL | 0 |
| **SwGateEn** | 1 = Enable SW-TOR | BOOL | 0 |
| `DelTiBad` | Delay time for output Bad [s] | REAL | 0.0 |
| `SampleTime` | Sample Time [s] | REAL | 0.1 |
| **TextRef** | Text reference | WORD | 16#0000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### In/out parameters

| Parameter | Meaning | Type | Default |
|---|---|---|---|
| **DataXchg** | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| `DataXchg1` | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| `MS_Xchg` | Exchange of the maintenance status | DWORD | 16#00000000 |
| **Mode** | Operating mode | DWORD | 16#00000000 |

### Output parameters

| Parameter | Meaning | Type | Default |
|---|---|---|---|
| **Bad** | 1 = invalid values | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **CmpVaL0** | 1 = comparison value 1 | BOOL | 0 |
| CmpVal1 | 1 = comparison value 2 | BOOL | 0 |
| CmpVal2 | 1 = comparison value 3 | BOOL | 0 |
| CmpVal3 | 1 = comparison value 4 | BOOL | 0 |

| Parameter | Meaning | Type | Default |
|---|---|---|---|
| **CntDir** | Direction counter | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| **CntRun** | 1 = status counter working | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| CntSync | 1 = status counter synchronized | BOOL | 0 |
| ErrorNum | Parameter error | INT | -1 |
| IntGate | 1 = status of internal TOR | BOOL | 0 |
| ModErr | 1 = higher-level error | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| **PV1** | Current load or LATCH value/current count value | DINT | 0 |
| **PV1CycLi** | Pulses per cycle | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| **PV1_Li** | Count value as REAL structure | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| **PV1_UnitLi** | Unit of the current count value | INT | 0 |
| **PV2** | Measured value | DINT | 0 |
| **PV2_Li** | Measured value as REAL structure | STRUCT | - |
| | | • Value: REAL | • 0.0 |
| | | • ST: BYTE | • 16#80 |
| **PV2_UnitLi** | Unit of the measured value | INT | 0 |
| NewLatch | 1 = new LATCH value (in clock-synchronous mode only) | BOOL | 0 |
| OpErr | 1 = operator error | BOOL | 0 |
| OosAct | Field device out of service, maintenance | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| **OFlow** | 1 = status overflow | BOOL | 0 |
| RstSync | 1 = Reset synchronization | STRUCT | - |
| | | • Value: BOOL | • 0 |
| | | • ST: BYTE | • 16#80 |
| **ScalePV1_Li** | Range of current counter values | STRUCT | - |
| | | • High: REAL | • 100.0 |
| | | • Low: REAL | • 0.0 |

| Parameter | Meaning | Type | Default |
|---|---|---|---|
| **ScalePV2_Li** | Range of the measured values | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| **SetDi** | 1 = status digital input DI set | BOOL | 0 |
| SimAct | 1 = simulation values | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **StartDI** | 1 = digital input DI start | BOOL | 0 |
| **StCmpV0** | 1 = Saved status of comparator 1 | BOOL | 0 |
| StCmpV1 | 1 = Saved status of comparator 2 | BOOL | 0 |
| StCmpV2 | 1 = Saved status of comparator 3; | BOOL | 0 |
| StCmpV3 | 1 = Saved status of comparator 4; | BOOL | 0 |
| **StopDI** | 1 = status digital input DI stop | BOOL | 0 |
| ST_Worst | Worst signal status | BYTE | 16#80 |
| SwGate | 1 = Software gate | BOOL | 0 |
| **UFlow** | 1 = status underflow | BOOL | 0 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |
| **ZeroSt** | 1 = status zero crossing | BOOL | 0 |

## Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| Laddr | IN | | | x |
| Channel | IN | | | x |
| Mode | IN / OUT | x | | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| MS_Xchg | IN / OUT | x | | |
| MS | IN | x | | |
| FM_DATA | IN / OUT | x | | |

## See also

Pcs7Cnt1 functions (Page 2627)

Parameters that cannot be connected (Page 75)

## 17.17.7 Pcs7Cnt1 block diagram

### Block diagram of Pcs7Cnt1

A block diagram is not provided for this block.

# 17.18 Pcs7Cnt2 - Control and read an 8-DI_NAMUR module of the ET 200iSP

## 17.18.1 Description of Pcs7Cnt2

### Object name (type + number)

Type + number: FB 1834

Family: Channel

### Area of application

The Pcs7Cnt2 block is used to control and read a count or frequency value from an 8-DI NAMUR module of the ET 200iSP. The block supports the following configurations of the module:

- 2 counters or 1 counter cascaded
- 2 frequency measurements

### How it works

Depending on the mode setting of the module in HW Config, the user data of the module are stored in the process image. The Pcs7Cnt2 block differentiates between the following modes:

| MODE (low word) | Operating mode | HW Config setting: "Configuration" | HW Config setting: "Channel (0 to 1) mode" |
|---|---|---|---|
| 1 | Counter (16 bits) without control function by means of digital signals | (Channel 0 to 1): COUNT(Channel 2 to 7): DI | Periodic or normal count functions (up or down counter) |
| 2 | Counter (32 bits) without control function by means of digital signals | (Channel 0 to 1): COUNT (Channel 2 to 7): DI | Cascade function (channel 0 only) (down counter) |
| 3 | Counter (16 bits) with control function by means of digital signals | (Channel 0 to 1): COUNT (Channel 2 to 7): CONTROL | Periodic or normal count functions (up or down counter) |
| 4 | Counter (32 bits) with control function by means of digital signals | (Channel 0 to 1): COUNT (Channel 2 to 7): CONTROL | Cascade function (channel 0 only) (down counter) |
| 5 | Frequency (16 bits) | (Channel 0 to 1): TRACE (Channel 2 to 7): DI | - |

The driver generator sets the module mode configured in HW Config at the `MODE` input of the `MOD_D1` block on the appropriate channel of the module. The `Mode` input indicates in what format the count or frequency value is available in the process image. If the high word of the input parameter `MODE = 16#40xxxx` (value status = higher-level error, `ModErr = 1`), the count or frequency value is treated as invalid.

Depending on the mode, either two independent counters (16 bits) or one counter (32 bits) exist in the process image. The `Channel` input specifies the module counter for which the block is responsible.

The counter functions can be controlled by signals that can be influenced both over the digital inputs of the module or over the user data of the process image.

**Note**

Please note that the signals of the digital inputs are ORed with the equivalent signals from the PIO in the module.

The following signals are available:

| Block input | Module | Meaning |
|---|---|---|
| - | Z1 | Counter pulse counter 1 |
| - | Z2 | Counter pulse counter 2 |
| `StopGate (Channel = 0)` | TOR1 | With the active GATE signal, an active count operation can be interrupted.<br>The GATE = "1" signal stops the count operation despite pending count pulses. At the same time, the assigned output is deactivated if it was active. This state remains until the GATE signal is set to "0". The output is brought to the previous state and the count operation is continued.<br>The GATE signal is subordinate to the RSO and RSC signals, in other words, the RSO and RSC signals have the effect described above regardless of an active GATE signal. |
| `StopGate (Channel = 1)` | TOR2 | See description of "GATE 1" |
| `RstPV1 (Channel = 0)` | RSZ1 | The rising edge of the RSC signal sets the count of the assigned channel as follows:<br>• When counting up (normal counter function), back to zero<br>• When counting down (periodic counter function and cascade function), to the defined setpoint<br>When counting down (periodic counter function and cascade function), any output that is set is also reset. |
| `RstPV1 (Channel = 1)` | RSZ2 | See description of "RSC1" |
| `RstDO (Channel = 0)` | RSA1 | On the rising edge of the RSO signal, the assigned output can be reset.<br>The count is not influenced by setting RSO. |
| `Rst_DO (CHANNEL = 1)` | RSA2 | See description of "RSO2" |

The in/out parameters `RstPV1` and `RstDO` are always reset to zero. After resetting, the earliest point at which a renewed reset will be possible is in the next cycle but one (rising edge).

The count value or frequency value and their states are stored in the process image as shown below and are indicated at the following block outputs:

| Bytes | Bit | Input signal | Block output | Meaning |
|---|---|---|---|---|
| 0, 1 | 0-15 | Proc. value counter 1 | `PV1_Li` | 16-bit counter 1 or 32-bit counter (bytes 0 to 3) or frequency value 1 |
| 2, 3 | 0-15 | Proc. value counter 2 | | 16-bit counter 2 (only with 16-bit counter 1) or frequency value 2 |
| 4 | 0 | A 1 | `ZeroSt` | Zero crossing counter 1 |
| | 1 | A2 | | Zero crossing counter 2 |

| Bytes | Bit | Input signal | Block output | Meaning |
|---|---|---|---|---|
| | 2 | TOR 1 | PV1 | Status gate 1 |
| | 3 | TOR 2 | | Status gate 2 |
| | 4 | RSZ1 | SimAct | Status reset counter 1 |
| | 5 | RSZ2 | | Status reset counter 2 |
| | 6 | RSA1 | RstDO_Out | Status reset outputs counter 1 |
| | 7 | RSA2 | | Status reset outputs counter 2 |

The LoadPV1 parameter is always written to the process image. Depending on the mode set with HW Config, it is either the 16-bit or 32-bit setpoint (down counter) or the count limit (up counter).

Depending on the mode setting, only the following integer values of LoadPV1 or LoadPV1_Li are transferred to the module based on the status at LoadPV1_Li:

- 16-bit counter: 0 to 65 535
- 32-bit counter: 0 to 2 080 374 784

If the value for LoadPV1 or LoadPV1_Li is outside of these limits, the last valid value of LoadPV1 or LoadPV1_Li is retained in the module and OpErr = 1 is set.

The block also supplies the counted pulses of PV1(PV1_Li) at the PV1CycLi output with each block call.

The PV1_UnitLi and ScalePV1_Li outputs are written with the inputs of the same name PV1_Unit and ScalePV1.

The LoadPV1 input parameter is the relevant value which is passed to the module. If, however, LoadPV1_Li is interconnected or the status of LoadPV1_Li is 16#80 , this value is written to LoadPV1 and applied.

**Note**

Make sure that no values with zero are loaded with the "Down counter" counter function.

## Configuration

When the CFC function "**Generate module drivers**" is used, the following occurs automatically:

- The Laddr, Laddr1, Channel inputs are configured.
- The Mode input is interconnected with the OMODEx output of the MOD_D1 block.
- The DataXchg input is interconnected to the relevant DXCHG_00 output of the MOD_D1 block.
- The MS input is interconnected to the relevant O_MS output of the MOD_D1 block.

**Note**

In HW Config, it is possible to use DI2 to DI7 as independent digital input signals with the hardware configuration "(Channel 0 to 1): COUNT (Channel 2 to 7): DI. The addresses of the connections "TOR1 to RSA2 (Byte 4.2 - 4.7)" must be used.

The usage of the digital control signals with the HW Config "Channel 2...7 = CONTROL" is not permitted.

Example:

| Module | Block | Has the effect | |
|---|---|---|---|
| TOR 1 = 1 | StopGate = 0 | | TOR on |
| TOR 1 = 0 | StopGate = 1 | | TOR on |
| TOR 1 = 0 | StopGate = 0 | | TOR off |

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation

This block does not have the Status parameter.

### See also

Pcs7Cnt2 I/Os (Page 2642)

## 17.18.2 Pcs7Cnt2 modes

### Operating modes of Pcs7Cnt2

This block does not have any modes.

## 17.18.3 Pcs7Cnt2 functions

### Functions of Pcs7Cnt2

The functions for this block are listed below.

### Addressing

You must connect the icon (from the icon table) for the count or frequency value with the `Connect` input parameter.

### Simulation

The block provides the standard function Simulating signals (Page 64).

With `SimOn = 1`, the simulation value `SimPV1` is written to the outputs `PV1` and `PV1_Li`. The status of the outputs is set to 16#60 and `Bad = 1` in the process.

Simulation takes highest priority.

If the block is in simulation state, `SimAct = 1` is set.

### Flutter suppression

This block provides the standard function Flutter suppression for channel blocks (Page 73).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

You need to assign parameters for three `Feature` bits for the response to an invalid raw value for the channel blocks.

If more than one of these `Feature` bits are set (=1), the following priority applies:

- Output invalid raw value (`Feature.Bit28` = highest priority)
- Output substitute value if raw value is invalid (`Feature.Bit29`)
- Output the last valid value if raw value is invalid (`Feature.Bit30`, lowest priority)

| Bit | Function |
|---|---|
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |
| 28 | Output invalid raw value (Page 185) |
| 29 | Output substitute value if raw value is invalid (Page 160) |
| 30 | Outputting last valid value if raw value is invalid (Page 165) |

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for channel blocks for field devices (Page 127).

| Value | Meaning |
|---|---|
| 16#80 | Valid value |
| 16#60 | Manipulated value (for example, substitute value, simulation, last valid value) |
| 16#00 | Invalid value |

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `LoadPV1_Li.ST`
- `PV1_Li.ST`
- `RstDO.ST`
- `RstPV1.ST`

### Redundancy

The higher-level MOD_D1 block evaluates the redundancy of DP master systems operating in an H system.

### Delay the output "Bad"

If input `DelTiBad` > 0, on receiving an internal bad value, delay timer will start. Timer will abort if `DelTiBad` is 0. Minimum value of `DelTiBad` is 0. Only when timer expires i.e., `RemDelTiBad` = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running `Bad.ST` will be 16#60. Only after the timer expires the actual `ST` be updated. Substitute value is updated at `PV_Out`, only after delay timer expires. When timer is running last valid value will be there at `PV_Out`. When substitute value is used, `ST` of `PV_Out` will be 16#60.

## 17.18.4 Pcs7Cnt2 error handling

### Error handling of Pcs7Cnt2

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers
- Higher-level error

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. |
| 21 | Pulse counter overflow |

### Higher-level error

A higher-level error is displayed at the output parameters `ModErr` and `Bad` with 1 if the signal status in the high word of the input parameter `Mode` assumes the value 16#40.

In addition, at the output parameter `PV_Li` of the signal status, either 16#00 (in the event of an error) or 16#60 (for manipulated value (for example, substitute value, simulation, last valid value) is output.

## 17.18.5 Pcs7Cnt2 messaging

### Messaging

This block does not offer messaging.

## 17.18.6 Pcs7Cnt2 I/Os

I/Os of CH_CNT2

### Input parameters

| Parameter | Meaning | Type | Default |
|---|---|---|---|
| Channel | Channel 8 DI NAMUR (outputs) | INT | 0 |
| **Connect** | Connects values | ANY | |
| `Feature` | I/O for additional functions (Page 2639) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 28: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 0<br>• 0 |
| FlutTmIn* | Flutter suppression time | INT | 0 |
| Laddr | Logical input address | INT | 0 |
| Laddr1 | Logical output address | INT | 0 |
| LoadPV1* | Counter load value | DINT | 0 |

| Parameter | Meaning | Type | Default |
|---|---|---|---|
| **LoadPV1_Li** | Counter load value structured | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#00 |
| **MS** | Maintenance status | DWORD | 16#00000000 |
| **MS_Release** | Release for maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **MS_Ext** | External maintenance status. For more information, refer to the `MS_Ext` (Page 2463) parameter section under information on using channel blocks. | DWORD | 16#00000000 |
| **PV1_Unit** | Unit of the current value | INT | 0 |
| **RstDO*** | 1 = Reset digital outputs | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **RstPV1*** | 1 = Reset counter | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **ScalePV1** | Range of the current count value | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| **SimOn** | 1 = simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **SimPV1** | Simulation value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| **StopGate** | 1 = Stop gate | BOOL | 1 |
| **SubsPV1** | Substitute value | DINT | 0 |
| `DelTiBad` | Delay time for output Bad [s] | REAL | 0.0 |
| `SampleTime` | Sample Time [s] | REAL | 0.1 |
| **TextRef** | Text reference | WORD | 16#0000 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## In/out parameters

| Parameter | Meaning | Type | Default |
|---|---|---|---|
| **DataXchg** | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| **Mode** | Operating mode | DWORD | 16#00000000 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |

## Output parameters

| Parameter | Meaning | Type | Default |
|---|---|---|---|
| **Bad** | 1 = invalid values | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ErrorNum | Parameter error | INT | -1 |
| Gate | 1 = TOR on | BOOL | 0 |
| ModErr | 1 = higher-level error | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OosAct | Field device out of service, maintenance is taking place | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| OpErr | 1 = operator error | BOOL | 0 |
| **PV1** | Current count value/frequency value | DINT | 0 |
| **PV1CycLi** | Pulses per cycle | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| **PV1_Li** | Current count value structured | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| **PV1_UnitLi** | Unit of the current count value | INT | |
| RstDO_Out | 1 = Reset digital outputs | BOOL | 0 |
| RstPV1_Out | 1 = Reset counter | BOOL | 0 |
| **ScalePV1_Li** | Range of the current count value | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| SimAct | 1 = Simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal value | BYTE | 16#80 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |
| **ZeroSt** | 1 = status zero crossing | BOOL | 0 |

### Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| Laddr | IN | | | x |
| Laddr1 | IN | | | x |
| Channel | IN | | | x |
| Mode | IN / OUT | x | | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| MS_Xchg | IN / OUT | x | | |
| MS | IN | x | | |

### See also

Pcs7Cnt2 functions (Page 2639)

Parameters that cannot be connected (Page 75)

## 17.18.7 Pcs7Cnt2 block diagram

### Block diagram of Pcs7Cnt2

A block diagram is not provided for this block.

# 17.19 Pcs7Cnt3 - Control and read the 1 COUNT 24V/100kHz module for count mode

## 17.19.1 Description of Pcs7Cnt3

### Object name (type + number)

Type + number: FB 1835

Family: Channel

### Area of application

The block is used to control and read count, measured and latch values of the "1 COUNT 24V/ 100kHz" module (as of 6ES7 138-4DA04-0AB0) for the count and measurement modes.

### How it works

The block communicates via the process image. The data are written and read continuously.

In `Mode` 1- 3, the `LoadVal` parameter is sent to the module when a positive edge is detected either at the `LoadPre` in/out parameter (= load counter in preparation) or `LoadDir` (= load counter immediately). The parameters `CmpV1` (comparison value 1) and `CmpV2` (comparison value 2) are transferred to the module when they are changed and during startup.

In `Mode` 4- 6, the parameters `UFlowLi`(low limit) and `OFlowLi` (high limit) are transferred to the module when they are changed and during startup.

Depending on the mode setting of the module in HW Config, the user data of the module is stored in the process image. The block distinguishes between the following modes:

| MODE (LowWord) | Operating mode | Description |
|---|---|---|
| 1 | Continuous counting | The 1Count24V/100kHz counts continuously from the load value in this mode |
| 2 | One-time counting | In this mode, 1Count24V/100kHz counts once, depending on the configured main counting direction |
| 3 | Periodic counting | In this mode, 1Count24V/100kHz counts periodically, depending on the configured main counting direction |
| 4 | Frequency measurement | 1Count24V/100kHz determines the frequency of the pulse sequence at the input |
| 5 | Speed measurement | 1Count24V/100kHz determines the speed of the device connected to the input |
| 6 | Period duration measurement | 1Count24V/100kHz determines the pulse length of the pulse sequence at the input |

The Mode input indicates the format in which the count and latch value or the count and measured value is available in the process image. If the high byte of input/output parameter is Mode = 16#40 (value status = higher-level error, ModErr = 1), the count, measured and latch values are treated as invalid.

The states PV1_sync (synchronization), CmpVal0 (comparator 0), CmpVal1 (comparator 1), OFlow (overflow), Uflow (underflow) and ZeroSt (zero crossing) are acknowledged automatically by the block. They are active for at least one cycle.

The block also supplies the counted pulses of PV1(PV1_Li) at the PV1CycLi output with each block call.

The units for PV1_UnitLi, ScalePV1_Li, PV2_UnitLi and ScalePV2_Li are set with the PV1_Unit, ScalePV1, PV2_Unit and ScalePV2 inputs of the same name.

The LoadPV1 input parameter is the relevant value which is passed to the module. If, however, LoadPV1_Li is interconnected or the status of LoadPV1_Li is 16#80 , the value is written to LoadPV1 and applied.

## Calling OBs

OB100 and cyclic OB (recommendation 100 ms) in which the data will be received and sent.

## Configuration

Connect the symbol (from the symbol table) for the count value with the Connect input parameter. You need to enter the symbol (Symbol column) in the symbol table and the supplement the row in the Address column with the ED base address of the module (e.g. ED512). When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The Laddr input is configured.
- The Mode input is interconnected to the OMODE_00output of the MOD_D1 block.
- The DataXchg input is interconnected to the relevant DXCHG_00 output of the MOD_D1 block.
- The MS input is interconnected to the relevant O_MS output of the MOD_D1 block.

The block is also installed automatically in the startup OB (OB100).

## Startup characteristics

In Mode 1-3, the parameters CmpV1 (comparison value 1) and CmpV2 (comparison value 2) are transferred to the module during startup.

In Mode 4-6, the parameters UflowLi (low limit) and OFlowLi (high limit) are transferred to the module during startup .

## Status word allocation

This block does not have the Status parameter.

**See also**

Pcs7Cnt3 I/Os (Page 2650)

## 17.19.2 Pcs7Cnt3 modes

### Operating modes of Pcs7Cnt3

This block does not have any modes.

## 17.19.3 Pcs7Cnt3 functions

### Functions of Pcs7Cnt3

The functions for this block are listed below.

### Addressing

Connect the symbol (from the symbol table) for the count value with the `Connect` input parameter.

You need to enter the symbol (Symbol column) in the symbol table and the supplement the row in the Address column with the ED base address of the module (e.g. ED512).

### Simulation

The block provides the standard function Simulating signals (Page 64)

With `SimOn = 1` , the simulation values `SimPV1` and `SimPV2` are written to the outputs `PV1`, `PV1_Li` as well as `PV2` and `PV2_Li` . The status of the outputs is set to 16#60 and `Bad = 1` in the process.

Simulation takes highest priority.

If the block is in simulation state, `SimAct = 1` is set.

### Flutter suppression

This block provides the standard function Flutter suppression for channel blocks (Page 73).

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 10 | Condition monitoring information at the channel blocks (Page 155) |
| 27 | Do not evaluate the first run of the diagnostic blocks (Page 205) |
| 28 | Output invalid raw value (Page 185) |
| 29 | Output substitute value if raw value is invalid (Page 160) |
| 30 | Outputting last valid value if raw value is invalid (Page 165) |

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for channel blocks for field devices (Page 127).

| Value | Meaning |
|---|---|
| 16#80 | Valid value |
| 16#60 | Manipulated value (for example, substitute value, simulation, last valid value) |
| 16#00 | Invalid value |

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `LoadPV1_Li.ST`
- `PV1_Li.ST`
- `PV2_Li.ST`

### Delay the output "Bad"

If input `DelTiBad` > 0, on receiving an internal bad value, delay timer will start. Timer will abort if `DelTiBad` is 0. Minimum value of `DelTiBad` is 0. Only when timer expires i.e., `RemDelTiBad` = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running `Bad.ST` will be 16#60. Only after the timer expires the actual `ST` be updated. Substitute value is updated at `PV_Out`, only after delay timer expires. When timer is running last valid value will be there at `PV_Out`. When substitute value is used, `ST` of `PV_Out` will be 16#60.

## 17.19.4 Pcs7Cnt3 error handling

### Error handling of Pcs7Cnt3

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Higher-level error
- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; the block is not executed |
| 0 | There is no error. |
| 21 | Pulse counter overflow |

### Higher-level error

A higher-level error is displayed at the output parameters `ModErr` and `Bad` with 1 if the signal status in the high word of the input parameter `Mode` assumes the value 16#40.

In addition, at the output parameter `PV_Li` of the signal status, either 16#00 (in the event of an error) or 16#60 (for manipulated value (for example, substitute value, simulation, last valid value) is output.

## 17.19.5 Pcs7Cnt3 messaging

### Messaging

This block does not offer messaging.

## 17.19.6 Pcs7Cnt3 I/Os

I/Os of Pcs7Cnt3

### Input parameters

| Parameter | Meaning | Type | Default |
|---|---|---|---|
| **AckTrip** | 1 = Error detected | BOOL | 0 |
| **CmpV1** | Comparison value 1 | DINT | 0 |
| CmpV2 | Comparison value 2 | DINT | 0 |
| **Connect** | Connection value | DWORD | 16#00000000 |
| **CtrlDO1** | 1=Enable DO1 | BOOL | 1 |
| CtrlDO2 | 1=Enable DO2 | BOOL | 1 |

| Parameter | Meaning | Type | Default |
|---|---|---|---|
| CtrlSyncEn | 1=Enable synchronization | BOOL | 0 |
| Feature | I/O for additional functions (Page 2648) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 28: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 0<br>• 0 |
| FlutEn | 1 = Flutter suppression active | BOOL | 0 |
| FlutTmIn* | Flutter suppression time | INT | 0 |
| Laddr | Logic address of the module | INT | 0 |
| LoadPV1* | Load counter | DINT | 0 |
| **LoadPV1_Li** | Load counter structured | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#00 |
| **LoadDir** | 1 = Load counter directly | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **LoadPre** | 1 = Load prepared counter | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **MS** | Maintenance status | DWORD | 16#00000000 |
| **MS_Release** | Release for maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **MS_Ext** | External maintenance status. For more information, refer to the `MS_Ext` (Page 2463) parameter section under information on using channel blocks. | DWORD | 16#00000000 |
| **PV1_Unit** | Unit of the current value | INT | 0 |
| **PV2_Unit** | Unit of the measured value | INT | 0 |
| **OFlow*** | High limit | DINT | 0 |
| **OFlowLi** | High limit structured | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| **ScalePV1** | Range of the current count value | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| **ScalePV2** | Range of current measured value | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| **SetDO1** | 1= Open DO1 | BOOL | 0 |
| SetDO2 | 1= Open DO2 | BOOL | 0 |

| Parameter | Meaning | Type | Default |
|---|---|---|---|
| **SimPV1** | Simulation count value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| **SimPV2** | Simulation measured value/simulation latch value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| **SimOn** | 1= simulation on | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **SubsPV1** | Count substitute value | DINT | 0 |
| **SubsPV2** | Measured/latch substitute value | DINT | 0 |
| **SwGateEn** | 1= enable software gate | BOOL | 0 |
| **TextRef** | Text reference | WORD | 16#0000 |
| **UFlow*** | Low limit | DINT | 0 |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |
| SampleTime | Sample Time [s] | REAL | 0.1 |
| **UFlowLi** | Low limit structured | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

## In/out parameters

| Parameter | Meaning | Data type | Default |
|---|---|---|---|
| **DataXchg** | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| **Mode** | Operating mode | DWORD | 16#00000000 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |

## Output parameters

| Parameter | Meaning | Data type | Default |
|---|---|---|---|
| **Bad** | 1 = invalid process value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **CmpVal1** | 1 = status of comparator 1 | BOOL | 0 |
| CmpVal2 | 1 = status of comparator 2 | BOOL | 0 |
| **DO1St** | 1 = status DO1 | BOOL | 0 |
| **DO1Err** | 1= short circuit / wire break / overtemperature | BOOL | 0 |
| **DO2St** | 1 = status DO2 | BOOL | 0 |

| Parameter | Meaning | Data type | Default |
|---|---|---|---|
| ErrorNum | Error message | INT | -1 |
| IntGate | 1 = status of internal gate | BOOL | 0 |
| **LoadErr** | 1= error in load function | BOOL | 0 |
| ModErr | 1= higher-level error | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Dev | Maintenance state | DWORD | 16#00000000 |
| MS_Req | 1 = Maintenance request from the diagnostic area | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **OFlowSt** | 1 = high count value | BOOL | 0 |
| OosAct | Field device out of service, maintenance is taking place | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| **ParaErr** | 1= parameter assignment error | BOOL | 0 |
| **PV1** | Current count value | DINT | 0 |
| **PV1CycLi** | Pulses per cycle | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| **PV1_Down** | 1 = status direction down | BOOL | 0 |
| **PV1_Li** | Current count value structured | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| **PV1_UnitLi** | Unit of the current count value | INT | 0 |
| **PV1_Up** | 1 = status direction up | BOOL | 0 |
| PV1_Sync | 1 = Count synchronous | BOOL | 0 |
| **PV2** | Measured value/latch value | DINT | 0 |
| **PV2_Li** | Structured measured value/latch value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| **PV2_UnitLi** | Unit of the measured value | INT | 0 |
| **ScalePV1_Li** | Range of the current count value | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| **ScalePV2_Li** | Range of current measured values/latch values | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| **SetDI** | 1 = Set digital input | BOOL | 0 |
| SimAct | 1= simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

| Parameter | Meaning | Data type | Default |
|---|---|---|---|
| ST_Worst | Worst signal status | BYTE | 16#80 |
| **UFlowSt** | 1 = low count value | BOOL | 0 |
| **V24Err** | 1= short circuit sensor power supply | BOOL | 0 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |
| **ZeroSt** | 1= status zero crossing | BOOL | 0 |

### Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| Laddr | IN | | | x |
| Mode | IN / OUT | x | | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| MS_Xchg | IN / OUT | x | | |
| MS | IN | x | | |

### See also

Pcs7Cnt3 functions (Page 2648)

Parameters that cannot be connected (Page 75)

## 17.19.7 Pcs7Cnt3 block diagram

### Block diagram of Pcs7Cnt3

A block diagram is not provided for this block.

# 17.20 Pcs7Cnt4 - Handling counting functionality of TM modules

## 17.20.1 Description of Pcs7Cnt4

### Object name (type + number) and family

Type + number: FB 1836

Family: Channel

### Area of application

The Pcs7Cnt4 block is used for the following applications:

- Control and read a count value from an "Fast input and output and counter functions" module of the ET 200SP HA.

### How it works

Depending on the mode setting of the module in HW Config, the user data of the module is stored in the process image. The Pcs7Cnt4 block differentiates between the following modes:

| MODE (low word) | Operating mode | HW Config setting: "Counting range" |
|---|---|---|
| 1 | Counter range 0 to 4294967295 | 0 to +32 bits |
| 2 | Counter range -2147483648 to 2147483647 | -31 to +31 bits |

The driver generator sets the mode configured in HW Config at the MODE input of the MOD_D32_PN block of the appropriate channel of the module. The Mode input indicates in which format the count value is available in the process image.

If the high word of the input parameter MODE = 16#40xxxx (value status = higher-level error, ModErr = 1), the count or frequency value is treated as invalid.

Depending on the mode, either the range of the counter value is 0 to 4294967295 or -2147483648 to 2147483647.

### Load Value

The value at LoadPV1 is written to the process image as long MODE = xxxx0001 or MODE = xxxx0002 and simulation is not active.

In Mode 1 LoadValue is limited to 0 if LoadValue < 0 and to 4294967295 if LoadValue > 4294967295.

In Mode 2 LoadValue is limited to -2147483648 if LoadValue < -2147483648 and to 2147483647 if LoadValue > 2147483647.

The inputs LoadPrep and LoadDirect can be used to transfer the load value to the module.

If an invalid load value is written to the module the module sets the bit OT_ERR to 1 which is displayed at the block output OpErr. This Bit needs to be reset by the block input RstOpErr before another load value can be written.

## Compare Values

The values at CmpV1 and CmpV2 are written to the process image as long MODE = xxxx0001 or MODE = xxxx0002 and simulation is not active.

In Mode 1 CmpV1/2 is limited to 0 if CmpV1/2 < 0 and to 4294967295 if CmpV1/2 > 4294967295.

In Mode 2 CmpV1/2 is limited to -2147483648 if CmpV1/2 < -2147483648 and to 2147483647 if CmpV1/2 > 2147483647.

The inputs TCmpVal1 and TCmpVal2 can be used to transfer the compare values to the module.

If an invalid compare value is written to the module the module sets the bit OT_ERR to 1 which is displayed at the block output OpErr. This Bit needs to be reset by the block input RstOpErr before another compare value can be written.

**Digital block inputs to control the counter**

The following digital block inputs are available to control the counter:

The functionality of the counter module differs depending on the configuration in HW Config.

| Block Inputs | Description |
|---|---|
| SwGate<br>StopGate | With the active GATE signal, an active count operation can be interrupted despite pending count pulses. The Functionality depends on the configuration in HW Config. Please see the manual for more information (Output Bits “SW_GATE” and “GATE_STP”). |
| LoadDirect<br>LoadPrep<br>EnSetUp<br>EnSetDown | Transfer LoadValue to the module. Please see the manual for more information (Output Bits “L_DIRECT”, “L_PREPAR”, “EN_SET_UP”, “EN_SET_DN”). |
| TCmpVal1<br>TCmpVal2 | Transfer Compare value 1 (CmpVal1) / Compare value 2 (CmpVal2) to the module. Please see the manual for more information (Output Bits “T_CMP_V1”and “T_CMP_V2”). |
| RstSync<br>RstZero<br>RstOpErr | Reset status bits SyncSt, ZeroSt and OpErr. Please see the manual for more information (Output Bits “RES_SYNC”, “RES_ZERO” and “OT_ERR_A”). |
| EnDQ0 E<br>nDQ1 | Enable analog output DQ0 / DQ1. Please see the manual for more information (Output Bits “ENABLE_DQ0” and “ENABLE_DQ1”) |

The Count value (Real) and its status is output at the block output PV1_Li. The Count value is also output as DINT value at the block output PV1. Attention: In Mode 1 the output is only valid for count values in the range of 0 to 2147483647.

The Latch/Load value (Real) and its status is output at the block output PV2_Li. Depending on the modules configuration the latch value or the load value is output. The Latch/Load value is also

output as DINT value at the block output PV2. Attention: In Mode 1 the output is only valid for Latch/Load values in the range of 0 to 2147483647.

The outputs PV1_UnitLi and ScalePV1_Li forward the values of the inputs PV1_Unit and ScalePV1.

## Configuration

Connect the DWord address of input "LATCH_LOAD" of the respective counter channel to the "Connect" input (e.g. ID13). The addressing can also be done via symbols by using the symbol table.

For example the "LATCH_LOAD" input of counter channel 1 has an offset of 13 to the base address of the module. If the input address range of the module starts at 512 the address ID525 needs to be connected to the Connect input to use the first counter channel.

The table below shows the offsets of the respective counter channel

| LATCH_LOAD value of counter channel x | Offset to base address |
|---|---|
| 1 | 13 |
| 2 | 24 |
| 3 | 35 |
| 4 | 46 |

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The Laddr and LaddrOut inputs are configured.
- The Mode input is interconnected to the OMODE_00output of the MOD_D32_PN block.
- The DataXchg input is interconnected to the relevant DXCHG_00 output of the MOD_D32_PN block.
- The DataXchg1 input is interconnected to the relevant DXCHG1_00 output of the MOD_D32_PN block.
- The MS input is interconnected to the relevant O_MS output of the MOD_D32_PN block.
- The ProImQB input is interconnected with the symbol of the counter's quality bit.

## Startup characteristics

The block does not have any startup characteristics.

## Redundancy

The block does not support redundancy.

## Status word allocation

This block does not have the Status parameter.

## 17.20.2 Pcs7Cnt4 modes

### Pcs7Cnt4 modes

This block does not have any modes.

## 17.20.3 Pcs7Cnt4 functions

### Simulation

The block provides the standard function Simulating signals.

With SimOn = 1 , the simulation values SimPV1 and SimPV2 are written to the outputs PV1, PV1_Li as well as PV2 and PV2_Li . The status of the outputs is set to 16#60 and the Bad output is set to 0.

In simulation the block is not reading nor writing from/to the process image.

Simulation takes highest priority.

If the block is in simulation state, SimAct = 1 is set.

### Flutter suppression

This block provides the standard function Flutter suppression for channel blocks.

### Configurable reactions using the Feature parameter

You can find an overview of all reactions provided by the Feature parameter in the Configurable functions using the Feature I/O. The following functionality is available for this block at the relevant bits:

You need to assign parameters for three Feature bits for the response to an invalid raw value for the channel blocks.

If more than one of these Feature bits are set (=1), the following priority applies:

Output invalid raw value (Feature.Bit28 = highest priority)

Output substitute value if raw value is invalid (Feature.Bit29)

Output the last valid value if raw value is invalid (Feature.Bit30, lowest priority)

| Bit | Function |
|---|---|
| 10 | Condition monitoring information at the channel blocks |
| 27 | Do not evaluate the first run of the diagnostic blocks |
| 28 | Output invalid raw value |
| 29 | Output substitute value if raw value is invalid |
| 30 | Outputting last valid value if raw value is invalid |

### Signal status for PCS7 channel blocks

The block provides the standard function Forming and outputting the signal status for PCS 7 channel blocks.

The worst signal status ST_Worst for the block is formed from the following parameters:

- LoadPV1.ST
- PV1_Li.ST
- PV2_Li.ST
- LoadDirect.ST
- LoadPrep.ST
- TCmpVal1.ST
- TCmpVal2.ST
- PV2_Li.ST
- RstSync.ST
- RstZero.ST
- RstOpErr.ST

### Delay the output "Bad"

If input `DelTiBad` > 0, on receiving an internal bad value, delay timer will start. Timer will abort if `DelTiBad` is 0. Minimum value of `DelTiBad` is 0. Only when timer expires i.e., `RemDelTiBad` = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running `Bad.ST` will be 16#60. Only after the timer expires the actual `ST` be updated. Substitute value is updated at `PV_Out`, only after delay timer expires. When timer is running last valid value will be there at `PV_Out`. When substitute value is used, `ST` of `PV_Out` will be 16#60.

### Unsigned DINT

Depending on your configuration, count channel of the module provides count values from 0 up to 4294967295. This corresponds to a data type unsigned DINT. Because STEP 7 does not provide a data type unsigned DINT, a misinterpretation of the highest bit takes place with STEP 7 data type DINT. That means, `PV_OutCnt` delivers values from 0...2147483647 and then goes on with -2147483648..-1. The count value is provided in parallel by output parameter `PV_Out` in data type REAL.

## 17.20.4 Pcs7Cnt4 error handling

### PCS7Cnt4 error handling

Refer to the section Error handling in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Channel error
- Higher-level error

A higher-level error is displayed at the output parameters ModErr and Bad with 1 if the signal status in the high word of the input parameter Mode assumes the value 16#40.

In addition, at the outputs PV1_Li and PV2_Li of the signal status, either 16#00 (in the event of an error) or 16#60 (for manipulated value (for example, substitute value, simulation, last valid value) is output.

## 17.20.5 Pcs7Cnt4 messaging

### PCS7Cnt4 messaging

This block does not offer messaging.

## 17.20.6 Pcs7Cnt4 I/Os

### Input parameters

| *Parameters* | *Description* | *Type* | *Default* |
|---|---|---|---|
| Connect | Connect to Latch/Load address of counter channel x | DWORD | 16#00000000 |
| CmpV1 | Compare Value 1 | REAL | 0.0 |
| CmpV2 | Compare Value 2 | REAL | 0.0 |
| EnDQ0 | 1 = Enable Digital output DQ0 | BOOL | 0 |
| EnDQ1 | 1 = Enable Digital output DQ1 | BOOL | 0 |
| EnSetDown | 1 = Enable Set Down | BOOL | 0 |
| EnSetUp | 1 = Enable Set Up | BOOL | 0 |
| Feature | I/O for additional functions | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 28: BOOL<br>• ...<br>• Bit 31: BOOL | <br>• 0<br>• 0<br>• 1<br>• 0<br>• 0 |
| FlutEn | 1=Fluttersupress active | BOOL | 0 |
| FlutTmIn | Fluttersupress time | INT | 0 |
| Laddr | Logical input address of counter x LatchLoad | INT | 0 |
| LaddrOut | Logical output address of counter x LatchValue | INT | 0 |

| Parameters | Description | Type | Default |
|---|---|---|---|
| LoadDirect | Rising edge: Transfer load value to module and apply load value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| LoadPrep | Rising edge: Transfer load value to module; Falling edge: Apply load value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| LoadPV1 | Load value | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| LoadPV1D | Load value (DINT) | STRUCT<br>• Value: DINT<br>• ST: BYTE | <br>• 0<br>• 16#80 |
| MS | Maintenance state | DWORD | 16#00000000 |
| MS_Ext | External Maintenance state | DWORD | 16#00000000 |
| MS_Release | Maintenance release | STRUCT<br>• Value: BOOL<br>• ST: BYTE | <br>• 0<br>• 16#80 |
| ProImQB | Qualitybit from Processimage | BOOL | 0 |
| PV1_Unit | Unit of counter value | INT | 0 |
| PV2_Unit | Unit of LatchLoad value | INT | 0 |
| RstOpErr | 1 = Acknowledge OT_ERR | STRUCT<br>• Value: BOOL<br>• ST: BYTE | <br>• 0<br>• 16#80 |
| RstSync | 1 = Reset Sync Status Bit (STS_SYNC) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | <br>• 0<br>• 16#80 |
| RstZero | 1 = Reset Status Bits STS_ZERO, STS_COMP1 and STS_COMP2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | <br>• 0<br>• 16#80 |
| StopGate | 1 = Close gate | BOOL | 0 |
| SwGate | Software gate; 1 = open; 0 = close | BOOL | 0 |
| TCmpVal1 | Rising edge: Transfer compare value 1 to module; Falling edge: Apply compare value 1 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | <br>• 0<br>• 16#80 |
| TCmpVal2 | Rising edge: Transfer compare value 2 to module; Falling edge: Apply compare value 2 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | <br>• 0<br>• 16#80 |
| ScalePV1 | Range of counter value | STRUCT<br>• High: REAL<br>• Low: REAL | <br>• 100.0<br>• 0.0 |

| *Parameters* | *Description* | *Type* | *Default* |
|---|---|---|---|
| ScalePV2 | Range of LatchLoad value | STRUCT<br>• High: REAL<br>• Low: REAL | <br>• 100.0<br>• 0.0 |
| SelQB | 1 = Select Qualitybit from Processimage | BOOL | 0 |
| SimOn | 1=Simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| SimPV1 | Simulation value Counter | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| SimPV2 | Simulation value LatchLoad | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0.0<br>• 16#80 |
| SubsPV1 | Substitution value Counter | REAL | 0.0 |
| SubsPV2 | Substitution value LatchLoad | REAL | 0.0 |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |
| SampleTime | Sample Time [s] | REAL | 0.1 |
| TextRef | Text reference external messages | WORD | 16#0000 |

## In/out Parameters

| *Parameter* | *Meaning* | *Type* | *Default* |
|---|---|---|---|
| DataXchg | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| Mode | Operating mode | DWORD | 16#00000000 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |

## Output Parameters

| *Parameter* | *Meaning* | *Type* | *Default* |
|---|---|---|---|
| Bad | 1 = bad process values | STRUCT<br>• Value: BOOL<br>• ST: BYTE | <br>• 0<br>• 16#80 |
| CntDir | Count Direction 0 = up 1 = down | BOOL | 0 |
| Comp1St | Status compare 1 | BOOL | 0 |
| Comp2St | Status compare 2 | BOOL | 0 |

| Parameter | Meaning | Type | Default |
|---|---|---|---|
| DiSetSt | Status of DI-Set | BOOL | 0 |
| DiStartSt | Status Di-Start | BOOL | 0 |
| DiStopSt | Status Di-Stop | BOOL | 0 |
| Dq0St | Status digital output 1 | BOOL | 0 |
| Dq1St | Status digital output 2 | BOOL | 0 |
| ErrorNum | Parameter error | INT | 0 |
| GateSt | Status gate; 1 = Gate open | BOOL | 0 |
| LoadDirectSt | Status of input LoadDirect | BOOL | 0 |
| LoadPrepSt | Status of input LoadPrep | BOOL | 0 |
| ModErr | 1=Higher level failure | STRUCT<br>• Value: BOOL<br>• ST: BYTE | <br>• 0<br>• 16#80 |
| MS_Dev | Maintenance State | DWORD | 16#00000000 |
| MS_Req | Maintenance request | STRUCT<br>• Value: BOOL<br>• ST: BYTE | <br>• 0<br>• 16#80 |
| OpErr | 1 = Operator Error | BOOL | 0 |
| OverflowSt | Status overflow: 1 = High counting limit reached | BOOL | 0 |
| PV1 | Counter value - Not valid for Values > 2147483647 (Limit of DInt) | DINT | 0 |
| PV1_Li | Counter value struct | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0<br>• 16#80 |
| PV1_UnitLi | Unit of actual value | INT | 0 |
| PV2 | Latch/Load value - Not valid for Values > 2147483647 (Limit of DInt) | DINT | 0 |
| PV2_Li | Latch/Load value struct | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0<br>• 16#80 |
| PV2_UnitLi | Unit of latch/load value | INT | 0 |
| RstSyncSt | Status of input RstSync | BOOL | 0 |
| RstZeroSt | Status of input RstZero | BOOL | 0 |
| RunSt | 1 = Counter is activated in HW Config | BOOL | 0 |
| ScalePV1_Li | Range of counter value | STRUCT<br>• High: REAL<br>• Low: REAL | <br>• 100.0<br>• 0.0 |
| ScalePV2_Li | Range of LatchLoad value | STRUCT<br>• High: REAL<br>• Low: REAL | <br>• 100.0<br>• 0.0 |

| *Parameter* | *Meaning* | *Type* | *Default* |
|---|---|---|---|
| SimAct | 1= Simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | <br>• 0<br>• 16#80 |
| ST_Worst | Worst Signal Status | BYTE | 0 |
| SwGateSt | Status software gate; 1 = Sw gate open | BOOL | 0 |
| SyncSt | Status synchronisation; 1 = Synchronisation was successful | BOOL | 0 |
| TCmpVal1St | Status of input TCmpVal1 | BOOL | 0 |
| TCmpVal2St | Status of input TCmpVal2 | BOOL | 0 |
| UnderflowSt | Status underflow: 1 = Low counting limit reached | BOOL | 0 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |
| ZeroSt | 1 = Zero crossing detected | BOOL | 0 |

## 17.20.7 Pcs7Cnt4 block diagram

### PCS7Cnt4 block diagram

A block diagram is not provided for this block.

# 17.21 Pcs7HaAI - HART variable channel block for AI-HART modules

## 17.21.1 Description of Pcs7HaAI

### Object name (type + number) and family

Type + number: FC 1931

Family: Channel

### Area of application for Pcs7HaAI

The block is used for providing `MultiHART` variables of an input HART field device. `MultiHART` is a special HART mode supported by ET 200SP HA modules.

The block supports 1 `MultiHART` and 4 `MultiHART`.

### How it works

Pcs7HaAI block provides the following HART variables that can be selected at input and output parameters.

Input parameters:

- `HV_PriEn` = 1 for the primary HART variable
- `HV_SecEn` = 1 for the secondary HART variable
- `HV_TerEn` = 1 for the tertiary HART variable
- `HV_QuaiEn` = 1 for the quaternary HART variable

Output parameters:

- `HV_OutP` primary HART variable including status
- `HV_OutS` secondary HART variable including status
- `HV_OutT` tertiary HART variable including status
- `HV_OutQ` quaternary HART variable including status

The process value is available as a physical variable. The signal status contains information about the status of the field device.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Connect the symbol generated in HW Config (symbol table) for the input channel with the `PV` input parameter.

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameter `MultiHART` is interconnected with the corresponding output parameter `MULTI_HART_xx` of the MHA_CO or OR_MHA_CO block.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

Pcs7HaAI modes (Page 2666)
Pcs7HaAI functions (Page 2666)
Pcs7HaAI error handling (Page 2667)
Pcs7HaAI messaging (Page 2668)
Pcs7HaAI I/Os (Page 2668)
Pcs7HaAI block diagram (Page 2671)

## 17.21.2 Pcs7HaAI modes

### Pcs7HaAI operating modes

This block does not have any operating modes.

### See also

Description of Pcs7HaAI (Page 2665)
Pcs7HaAI functions (Page 2666)
Pcs7HaAI error handling (Page 2667)
Pcs7HaAI messaging (Page 2668)
Pcs7HaAI I/Os (Page 2668)
Pcs7HaAI block diagram (Page 2671)

## 17.21.3 Pcs7HaAI functions

### Functions of Pcs7HaAI

The functions for this block are listed below.

### Obtaining the standard value

The HART variable value of the process image (partition) is outputted as a standard value at the HV_OutP/HV_OutS/HV_OutT/HV_OutQ output parameters.

### See also

Description of Pcs7HaAI (Page 2665)
Pcs7HaAI modes (Page 2666)
Pcs7HaAI error handling (Page 2667)
Pcs7HaAI messaging (Page 2668)
Pcs7HaAI I/Os (Page 2668)
Pcs7HaAI block diagram (Page 2671)
Configurable functions using the Feature I/O (Page 139)
Do not evaluate the first run of the diagnostic blocks (Page 205)

## 17.21.4 Pcs7HaAI error handling

### Error handling of Pcs7HaAI

The following errors can be displayed for this block:

- Invalid value

### Invalid value

The invalid values are displayed with the value 1 at the output parameters HV_OutPBad/HV_OutSBad/HV_OutTBad/HV_OutQBad. The invalid value error is generated from the signal status HV_PriQC/HV_SecQC/HV_TerQC/HV_QuaQC of the MultiHART in/out parameter when the signal status is 16#00 (initial value from S7- System), 16#37 (initial value from analog module after module start), 16#23 (Communication error or HART-variable in field device not available), or 16#28, 16#29, 16#2A, 16#2B (Bad from HART field device) respectively.

### See also

Description of Pcs7HaAI (Page 2665)
Pcs7HaAI modes (Page 2666)
Pcs7HaAI functions (Page 2666)
Pcs7HaAI messaging (Page 2668)
Pcs7HaAI I/Os (Page 2668)
Pcs7HaAI block diagram (Page 2671)

## 17.21.5 Pcs7HaAI messaging

### Messaging

This block does not offer messaging.

### See also

Description of Pcs7HaAI (Page 2665)
Pcs7HaAI modes (Page 2666)
Pcs7HaAI functions (Page 2666)
Pcs7HaAI error handling (Page 2667)
Pcs7HaAI I/Os (Page 2668)
Pcs7HaAI block diagram (Page 2671)

## 17.21.6 Pcs7HaAI I/Os

### I/Os of Pcs7HaAI

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional features | STRUCT<br>• Bit0: BOOL<br>• ...<br>• Bit28: BOOL<br>• ...<br>• Bit31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 0<br>• 0 |
| HV_PriEn | 1= Read primary HART variable | BOOL | 0 |
| HV_SecEn | 1= Read secondary HART variable | BOOL | 0 |
| HV_TerEn | 1= Read tertiary HART variable | BOOL | 0 |
| HV_QuaEn | 1= Read quaternary HART variable | BOOL | 0 |
| PV | Process value (analog value) | WORD | 16#0000 |
| HV_UnitP | Unit of primary HART variable | INT | 1001 |
| ScaleP | Range of primary HART variable | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| HV_UnitS | Unit of secondary HART variable | INT | 1001 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ScaleS | Range of secondary HART variable | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| HV_UnitT | Unit of tertiary HART variable | INT | 1001 |
| ScaleT | Range of tertiary HART variable | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| HV_UnitQ | Unit of quaternary HART variable | INT | 1001 |
| ScaleQ | Range of quaternary HART variable | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |

## In/out parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| MultiHART | Internal data exchange channel to the diagnostics block | STRUCT<br>• HV_PriEn: BOOL<br>• HV_SecEn: BOOL<br>• HV_TerEn: BOOL<br>• HV_QuaEn: BOOL<br>• HV_PriQC: BYTE<br>• HV_SecQC: BYTE<br>• HV_TerQC: BYTE<br>• HV_QuaQC: BYTE<br>• Command: BYTE<br>• Ack: BYTE<br>• HV_PriVal: REAL<br>• HV_SecVal: REAL<br>• HV_TerVal: REAL<br>• HV_QuaVal: REAL | -<br>• 0<br>• 0<br>• 0<br>• 0<br>• 16#00<br>• 16#00<br>• 16#00<br>• 16#00<br>• 16#00<br>• 16#00<br>• 0.0<br>• 0.0<br>• 0.0<br>• 0.0 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number | INT | -1 |
| HV_OutP | Primary HART variable with status | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| HV_OutS | Secondary HART variable with status | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| HV_OutT | Tertiary HART variable with status | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| HV_OutQ | Quaternary HART variable with status | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| HV_OutPBad | 1 = Bad primary HART variable | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| HV_OutSBad | 1 = Bad secondary HART variable | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| HV_OutTBad | 1 = Bad tertiary HART variable | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| HV_OutQBad | 1 = Bad quaternary HART variable | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| HV_OUniP | Unit of primary HART variable | INT | 0 |
| ScaleOuP | Range of primary HART variable | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| HV_OUniS | Unit of secondary HART variable | INT | 0 |
| ScaleOuS | Range of secondary HART variable | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| HV_OUniT | Unit of tertiary HART variable | INT | 0 |
| ScaleOuT | Range of tertiary HART variable | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| HV_OUniQ | Unit of quaternary HART variable | INT | 0 |
| ScaleOuQ | Range of quaternary HART variable | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |

### Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| MultiHart | IN / OUT | x | | |

### See also

Description of Pcs7HaAI (Page 2665)
Pcs7HaAI modes (Page 2666)
Pcs7HaAI functions (Page 2666)
Pcs7HaAI error handling (Page 2667)
Pcs7HaAI messaging (Page 2668)
Pcs7HaAI block diagram (Page 2671)
Parameters that cannot be connected (Page 75)

## 17.21.7 Pcs7HaAI block diagram

### Block diagram of Pcs7HaAI

A block diagram is not provided for this block.

### See also

Description of Pcs7HaAI (Page 2665)
Pcs7HaAI modes (Page 2666)
Pcs7HaAI functions (Page 2666)
Pcs7HaAI error handling (Page 2667)
Pcs7HaAI messaging (Page 2668)
Pcs7HaAI I/Os (Page 2668)

# 17.22 Pcs7HaAO - HART variable channel block for AO-HART modules

## 17.22.1 Description of Pcs7HaAO

### Object name (type + number) and family

Type + number: FC 1932

Family: Channel

### Area of application for Pcs7HaAO

The block is used for providing `MultiHART` variables of an output HART field device. `MultiHART` is a special HART mode supported by ET 200SP HA modules.

The block supports 1 `MultiHART` and 4 `MultiHART`.

### How it works

Pcs7HaAO block provides the following HART variables that can be selected at input and output parameters.

Input parameters:

- `HV_PriEn` = 1 for the primary HART variable
- `HV_SecEn` = 1 for the secondary HART variable
- `HV_TerEn` = 1 for the tertiary HART variable
- `HV_QuaiEn` = 1 for the quaternary HART variable

Output parameters:

- `HV_OutP` primary HART variable including status
- `HV_OutS` secondary HART variable including status
- `HV_OutT` tertiary HART variable including status
- `HV_OutQ` quaternary HART variable including status

The process value is available as a physical variable. The signal status contains information about the status of the field device.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Connect the symbol generated in HW Config (symbol table) for the output channel with the `PV_Out` output parameter.

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameter `MultiHART` is interconnected with the corresponding output parameter `MULTI_HART_xx` of the MHA_CO or OR_MHA_CO block.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

Pcs7HaAO modes (Page 2673)
Pcs7HaAO functions (Page 2673)
Pcs7HaAO error handling (Page 2674)
Pcs7HaAO messaging (Page 2675)
Pcs7HaAO I/Os (Page 2675)
Pcs7HaAO block diagram (Page 2678)

## 17.22.2 Pcs7HaAO modes

### Pcs7HaAO operating modes

This block does not have any operating modes.

### See also

Description of Pcs7HaAO (Page 2672)
Pcs7HaAO functions (Page 2673)
Pcs7HaAO error handling (Page 2674)
Pcs7HaAO messaging (Page 2675)
Pcs7HaAO I/Os (Page 2675)
Pcs7HaAO block diagram (Page 2678)

## 17.22.3 Pcs7HaAO functions

### Functions of Pcs7HaAO

The functions for this block are listed below.

### Obtaining the standard value

The HART variable value of the process image (partition) is outputted as a standard value at the `HV_OutP`/`HV_OutS`/`HV_OutT`/`HV_OutQ` output parameters.

### See also

Description of Pcs7HaAO (Page 2672)
Pcs7HaAO modes (Page 2673)
Pcs7HaAO error handling (Page 2674)
Pcs7HaAO messaging (Page 2675)
Pcs7HaAO I/Os (Page 2675)
Pcs7HaAO block diagram (Page 2678)
Configurable functions using the Feature I/O (Page 139)
Do not evaluate the first run of the diagnostic blocks (Page 205)

## 17.22.4 Pcs7HaAO error handling

### Error handling of Pcs7HaAO

The following errors can be displayed for this block:

- Invalid value

### Invalid value

The invalid values are displayed with the value 1 at the output parameters `HV_OutPBad`/ `HV_OutSBad`/`HV_OutTBad`/`HV_OutQBad`. The invalid value error is generated from the signal status HV_PriQC/HV_SecQC/HV_TerQC/HV_QuaQC of the `MultiHART` in/out parameter when the signal status is 16#00 (initial value from S7- System), 16#37 (initial value from analog module after module start), 16#23 (Communication error or HART-variable in field device not available), or 16#28, 16#29, 16#2A, 16#2B (Bad from HART field device) respectively.

### See also

Description of Pcs7HaAO (Page 2672)
Pcs7HaAO modes (Page 2673)
Pcs7HaAO functions (Page 2673)
Pcs7HaAO messaging (Page 2675)
Pcs7HaAO I/Os (Page 2675)
Pcs7HaAO block diagram (Page 2678)

## 17.22.5 Pcs7HaAO messaging

### Messaging

This block does not offer messaging.

### See also

Description of Pcs7HaAO (Page 2672)
Pcs7HaAO modes (Page 2673)
Pcs7HaAO functions (Page 2673)
Pcs7HaAO error handling (Page 2674)
Pcs7HaAO I/Os (Page 2675)
Pcs7HaAO block diagram (Page 2678)

## 17.22.6 Pcs7HaAO I/Os

### I/Os of Pcs7HaAO

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ChAdr | | INT | -1 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional features | STRUCT<br>• Bit0: BOOL<br>• ...<br>• Bit28: BOOL<br>• ...<br>• Bit31: BOOL | -<br>• 0<br>• 0<br>• 1<br>• 0<br>• 0 |
| HV_PriEn | 1= Read primary HART variable | BOOL | 0 |
| HV_SecEn | 1= Read secondary HART variable | BOOL | 0 |
| HV_TerEn | 1= Read tertiary HART variable | BOOL | 0 |
| HV_QuaEn | 1= Read quaternary HART variable | BOOL | 0 |
| HV_UnitP | Unit of primary HART variable | INT | 1001 |
| ScaleP | Range of primary HART variable | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| HV_UnitS | Unit of secondary HART variable | INT | 1001 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ScaleS | Range of secondary HART variable | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| HV_UnitT | Unit of tertiary HART variable | INT | 1001 |
| ScaleT | Range of tertiary HART variable | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| HV_UnitQ | Unit of quaternary HART variable | INT | 1001 |
| ScaleQ | Range of quaternary HART variable | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |

## In/out parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| MultiHART | Internal data exchange channel to the diagnostics block | STRUCT<br>• HV_PriEn: BOOL<br>• HV_SecEn: BOOL<br>• HV_TerEn: BOOL<br>• HV_QuaEn: BOOL<br>• HV_PriQC: BYTE<br>• HV_SecQC: BYTE<br>• HV_TerQC: BYTE<br>• HV_QuaQC: BYTE<br>• Command: BYTE<br>• Ack: BYTE<br>• HV_PriVal: REAL<br>• HV_SecVal: REAL<br>• HV_TerVal: REAL<br>• HV_QuaVal: REAL | -<br>• 0<br>• 0<br>• 0<br>• 0<br>• 16#00<br>• 16#00<br>• 16#00<br>• 16#00<br>• 16#00<br>• 16#00<br>• 0.0<br>• 0.0<br>• 0.0<br>• 0.0 |

## Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Output of pending error number | INT | -1 |
| HV_OutP | Primary HART variable with status | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| HV_OutS | Secondary HART variable with status | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| HV_OutT | Tertiary HART variable with status | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| HV_OutQ | Quaternary HART variable with status | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| HV_OutPBad | 1 = Bad primary HART variable | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| HV_OutSBad | 1 = Bad secondary HART variable | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| HV_OutTBad | 1 = Bad tertiary HART variable | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| HV_OutQBad | 1 = Bad quaternary HART variable | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| PV_Out | Process value | WORD | 16#0000 |
| HV_OUniP | Unit of primary HART variable | INT | 0 |
| ScaleOuP | Range of primary HART variable | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| HV_OUniS | Unit of secondary HART variable | INT | 0 |
| ScaleOuS | Range of secondary HART variable | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| HV_OUniT | Unit of tertiary HART variable | INT | 0 |
| ScaleOuT | Range of tertiary HART variable | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |
| HV_OUniQ | Unit of quaternary HART variable | INT | 0 |
| ScaleOuQ | Range of quaternary HART variable | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |

### Automatically interconnected and parameterized inputs

| Parameter name | Type | Driver wizard – Action at the time of compilation | | Parameterization |
|---|---|---|---|---|
| | | Connection with | | |
| | | Diagnostic block | Symbol | |
| DataXchg | IN / OUT | x | | |
| DataXchg1 | IN / OUT | x | | |
| MultiHart | IN / OUT | x | | |
| ChAdr | IN | | | x |

### See also

Description of Pcs7HaAO (Page 2672)
Pcs7HaAO modes (Page 2673)
Pcs7HaAO functions (Page 2673)
Pcs7HaAO error handling (Page 2674)
Pcs7HaAO messaging (Page 2675)
Pcs7HaAO block diagram (Page 2678)
Parameters that cannot be connected (Page 75)

## 17.22.7 Pcs7HaAO block diagram

### Block diagram of Pcs7HaAO

A block diagram is not provided for this block.

### See also

Description of Pcs7HaAO (Page 2672)
Pcs7HaAO modes (Page 2673)
Pcs7HaAO functions (Page 2673)
Pcs7HaAO error handling (Page 2674)
Pcs7HaAO messaging (Page 2675)
Pcs7HaAO I/Os (Page 2675)

# 17.23 Pcs7ReIn - REAL input channel block

## 17.23.1 Description of PCS7ReIn

### Object name (type + number) and family

Type + number: FB 1837

Family: Channel

### Area of application for Pcs7DiIT

The Pcs7ReIn block is used for the following applications:

- Signal processing of an analog input value of floating-point type (IEEE-754) from S7-300/400 SM analog input modules.
- Processing of the redundant input channels.

### How it works

The block cyclically processes all channel-specific signal functions of an analog input module.

It reads an analog value of floating-point type (IEEE-754) from the process image (partition) and outputs this value at PV_Out.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB(OB30 to OB38).

The block is also installed automatically in the startup OB(OB100).

Connect the symbol generated in the HW Config (symbol table) for the input channel with the `PV_In` input parameter. The symbol must be of datatype Real.

In case of redundant I/O configuration, use symbol of the channel of the master module.

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in parameter ProImQB is interconnected with the symbol of the quality bit.
- The in/out parameter Mode is interconnected to the corresponding OMode_xx output parameter of the MOD block.
- The in/out parameter DataXchg is interconnected to the corresponding DataXchg_xx output parameter of the MOD block.
- The in/out parameter DataXchg1 is interconnected to the corresponding DataXchg1_xx output parameter of the MOD block.
- The MS parameter is interconnected to the O_MS output parameter of the diagnostics block.

In case of redundancy, connection with both quality bits and connection with the redundant slave channel:

- The in parameter ProImQB is interconnected with the symbol of the quality bit of the master channel.
- The in parameter PV_InSlv is interconnected with the symbol of the slave channel.
- The in parameter ProImQBSlv is interconnected with the symbol of the quality bit of the slave channel.

If you are not using the CFC function "Generate module drivers" you must set the Mode in/out parameter manually. Byte 0 must be set to 16#01 for the block to output valid values. Byte 3 = 16#80 = "valid value". E.g. Mode = 16#80000001. Refer to the Mode Settings for SM Modules section for more on this.

### Startup characteristics

The block does not have any startup characteristics.

### Redundancy

The block monitors the values of the redundant signals (master and slave) with quality code and the value of the signal with “Good” quality code along with its quality code is transferred to the process.

In case both the signals deliver “Bad” quality code, the substitute value with “Bad” quality code is transferred to the process.

### Status word allocation

This block does not have the `Status` parameter.

## 17.23.2 PCS7ReIn modes

### PCS7ReIn modes

This block does not have any modes.

## 17.23.3 PCS7ReIn functions

### Functions of FbAnIn

The functions for this block are listed below.

### Obtaining the standard value

The analog value of the process image (partition) is output as a standard value at the PV_Li output parameter.

### Holding the last value if raw value is invalid

If the block is to hold its last valid value when the analog value is invalid or in the initialization phase of the device, you must activate this function at the Feature Bit Outputting last valid value if raw value is invalid.

### Output substitute value if raw value is invalid

If the block is to output a substitute value (SubsPV) when the analog value is invalid or in the initialization phase of the device, you must activate this function at the Feature Bit Output substitute value if raw value is invalid.

### Outputting an invalid value if analog value is invalid

If the block is to output an invalid value (PV_Li = PV), you must activate this function at the Feature Bit Output invalid raw value.

This function is preset.

### Flutter suppression

This block provides the standard function Flutter suppression for channel blocks

### Signal status for PCS7 channel blocks

The block provides the standard function Forming and outputting the signal status for PCS 7 channel blocks.

### Simulating signals

The block provides the standard function Simulating signals.

### Configurable reactions using the Feature parameter

You can find an overview of all reactions provided by the Feature parameter in the Configurable functions using the Feature I/O . The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 10 | Condition monitoring information at the channel blocks |
| 27 | Do not evaluate the first run of the diagnostic blocks |
| 28 | Output invalid raw value |

| Bit | Function |
|---|---|
| 29 | Output substitute value if raw value is invalid |
| 30 | Outputting last valid value if raw value is invalid |

### Sign-of-life monitoring

If an input value whose signal status is 16#80 (good) remains constant for a programmable time (monitoring time), the input value is detected as faulty and the outputs Bad = 1 and FrzVal = 1 are set.

The monitoring time is set at the FrznTmIn input parameter in seconds. With FrznTmIn = 0 or FrznEn = 0 (default setting), the sign-of-life monitoring is deactivated, any pending errors are reset.

The input value is considered as faulty as long as it seen as constant. The monitoring time is restarted each time the input value is changed.

### Delay the output "Bad"

If input `DelTiBad` > 0, on receiving an internal bad value, delay timer will start. Timer will abort if `DelTiBad` is 0. Minimum value of `DelTiBad` is 0. Only when timer expires i.e., `RemDelTiBad` = 0, the bad value will be updated to the output. If timer is running and internal bad value becomes 0, then whatever time is left will be shown in negative to bring to operators notice that bad value was received. During the time, delay timer is running `Bad.ST` will be 16#60. Only after the timer expires the actual `ST` be updated. Substitute value is updated at `PV_Out`, only after delay timer expires. When timer is running last valid value will be there at `PV_Out`. When substitute value is used, `ST` of `PV_Out` will be 16#60.

## 17.23.4 PCS7ReIn error handling

### Quintessence

Refer to the section Error handling in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Channel error
- Higher-level error
- Invalid measuring range
- Frozen input value (sign-of-life monitoring)

### Channel error

At the output parameter Bad, channel errors are displayed with 1.

### Higher-level error / invalid measuring range

A higher-level error is displayed at the output parameters ModErr and Bad with 1 if the signal status at High Word of the input parameter Mode accepts the value 16#40.

A higher-level error is also present when an incorrect measuring type is entered in Low Word of the input parameter Mode. Only 16#0001 is valid.

In addition, at the outputs PV1_Li and PV2_Li of the signal status, either 16#00 (in the event of an error) or 16#60 (for manipulated value (for example, substitute value, simulation, last valid value) is output.

## 17.23.5 PCS7ReIn messaging

### Messaaging

This block does not offer messaging.

## 17.23.6 PCS7ReIn I/Os

### Input parameters

| Parameters | Description | Type | Default |
|---|---|---|---|
| Deltaval | Delta value (PV_In - Last valid value) | REAL | 0.0 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| Feature | I/O for additional functions | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 28: BOOL<br>• ...<br>• Bit 31: BOOL | <br>• 0<br>• 0<br>• 1<br>• 0<br>• 0 |
| FlutEn | 1=Fluttersupress enable | BOOL | 0 |
| FlutTmIn | Fluttersupress time | INT | 0 |
| FrznEn | 1 = Sign-of-life monitoring activated | BOOL | 0 |
| FrznTmIn | Monitoring time in [s] | REAL | 0.0 |
| MS | Maintenance state | DWORD | 16#00000000 |
| MS_Ext | External Maintenance state | DWORD | 16#00000000 |

| Parameters | Description | Type | Default |
|---|---|---|---|
| MS_Release | Release for maintenance (interconnected with MS_Release of the technology block) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | <br>• 0<br>• 16#80 |
| ProImQB | Qualitybit from Processimage | BOOL | 0 |
| ProImQBSlv | Qualitybit from Processimage of slave channel | BOOL | 0 |
| PV_In | Input value | REAL | 0.0 |
| PV_InSlv | Input value of slave channel | REAL | 0.0 |
| PV_Unit | Unit of measure for process value | INT | 1001 |
| SampleTime | Sampling time in [s] | REAL | 0.1 |
| Scale | Range of process value | STRUCT<br>• High: REAL<br>• Low: REAL | <br>• 100.0<br>• 0.0 |
| SimOn | 1 = Simulation on | STRUCT<br>• Value: BOOL<br>• ST: BYTE | <br>• 0<br>• 16#80 |
| SimPV_In | Process value used for SimOn = 1 | STRUCT<br>• Value: REAL<br>• ST: BYTE | <br>• 0<br>• 16#80 |
| SubsPV_In | Substitute value | REAL | 0.0 |
| DelTiBad | Delay time for output Bad [s] | REAL | 0.0 |
| SampleTime | Sample Time [s] | REAL | 0.1 |
| TextRef | Text reference | WORD | 16#0000 |

## In/out Parameters

| Parameter | Meaning | Type | Default |
|---|---|---|---|
| DataXchg | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| DataXchg1 | Internal data exchange channel to the diagnostics block | DWORD | 16#00000000 |
| Mode | Operating mode | DWORD | 16#00000000 |
| MS_Xchg | Exchange of the maintenance status | DWORD | 16#00000000 |

## Output Parameters

| Parameter | Meaning | Type | Default |
|---|---|---|---|
| Bad | 1 = Process value is not valid | STRUCT | |
| | | Value: BOOL | 0 |
| | | ST: BYTE | 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ErrorNum | Parameter error | INT | 0 |
| FrznVal | Frozen process value | STRUCT | |
| | | Value: BOOL | 0 |
| | | ST: BYTE | 16#80 |
| ModErr | 1=Higher level failure | STRUCT | |
| | | Value: BOOL | 0 |
| | | ST: BYTE | 16#80 |
| MS_Dev | Maintenance State | DWORD | 16#00000000 |
| MS_Req | Maintenance request | STRUCT | |
| | | Value: BOOL | 0 |
| | | ST: BYTE | 16#80 |
| OosAct | 1 = Field device is undergoing maintenance | STRUCT | |
| | | Value: BOOL | 0 |
| | | ST: BYTE | 16#80 |
| PV_Out | Standard value (physical variable) | STRUCT | |
| | | Value: REAL | 0 |
| | | ST: BYTE | 16#80 |
| PV_OutUnit | Unit of the process value | INT | 0 |
| RemTime | Remaining monitoring time [s] | REAL | 0.0 |
| SampleTime | Sampling time [s] | REAL | 0.1 |
| ScaleOut | Range of process value | STRUCT | |
| | | High: REAL | • 100.0 |
| | | Low: REAL | • 0.0 |
| RemDelTiBad | Remaining delay time for output Bad [s] | REAL | 0.0 |
| SimAct | 1= Simulation active | STRUCT | |
| | | Value: BOOL | • 0 |
| | | ST: BYTE | • 16#80 |

## 17.23.7 PCS7ReIn block diagram

### PCS7ReIn block diagram

A block diagram is not provided for this block.

# 17.24 Annex for channel blocks

## 17.24.1 Mode Settings for SM Modules

### Mode Structure

The structure and meaning of the input Mode of data type DWORD are shown below:

| | | |
|---|---|---|
| Byte 3: | 16#80: Value status "valid value"<br>16#85 Value status "valid value"<br>16#00: Value status "invalid value"<br>16#40: Value status "invalid value" | <br>(IO redundancy warning)<br>(Channel error)<br>(Higher-level error) |
| Byte 2: | 16#01: Restart (OB 100) has been carried out<br>16#02: Measuring range high limit violated<br>16#04: Measuring range low limit violated<br>16#xy: Variant identification with multiple mode assignment (see below) | <br>(Channel-error diagnostics)<br>(Channel-error diagnostics) |
| Byte 1, 0 (low word): | Measuring range coding (see below) | |

Example:

16#80010203 = value status "valid value", restart has been carried out, current 4 mA to 20mA

### Mode: Celsius and Fahrenheit

| Variant | x | y | Unit of measurement |
|---|---|---|---|
| 0 | 0, 1<br>(depending on the other variant IDs) | 0,1,2,4 | Celsius |
| 1 | 8, 9<br>(depending on the other variant IDs) | 0,1,2,4 | Fahrenheit |

### Mode: 16#090C and variant

| Variant | x | y | Measuring range |
|---|---|---|---|
| 0 | 0 | 0,1,2,4 | Ni 120 standard range |
| 1 | 1 | 0,1,2,4 | KTY84/110 |

### Mode: 16#090D and variant

| Variant | x | y | Measuring range |
|---|---|---|---|
| 0 | 0 | 0,1,2,4 | Ni 120 climate range |
| 1 | 1 | 0,1,2,4 | KTY84/130 |

### Mode: 16#07 (Coding A) and variant

| Variant | x | y | Measuring range |
|---|---|---|---|
| 0 | 0 | 0,1,2,4 | HART interface |
| 1 | 1 | 0,1,2,4 | Thermocouple, dynamic reference temperature |

### MODE: 16#08, 16#0A, 16#0B, 16#0D, 16#0E (coding A) and variant

Table 17-1 PCS 7 as of V8.1: MODE parameter variant

| Variant | x | y | Measuring range |
|---|---|---|---|
| 0 | 0 | 0,1,2,4 | S7-300 module |
| 1 | 3 | 0,1,2,4 | S7-300, ET 200M Ex I/O module |

Table 17-2 PCS 7 V7.x: MODE parameter variant

| Variant | mode | Measuring range |
|---|---|---|
| 0 | 16#08, 16#0A,<br>16#0B, 16#0D,<br>16#0E | S7-300 module |
| 1 | 16#18, 16#1A,<br>16#1B, 16#1D,<br>16#1E | S7-300, ET 200M Ex I/O module |

### MODE: 1oo1-/1oo2 evaluation for F modules

| Variant | x | y | Measuring range |
|---|---|---|---|
| 0 | 0 | 0,1,2,4 | 1oo1 (1v1) evaluation |
| 1 | 4 | 0,1,2,4 | 1oo2 (2v2) evaluation |

### Measuring range coding of the analog input modules

Depending on the measuring range coding of the analog input modules, the parameter `Mode` (low word = measuring range coding) corresponding to the channel must be specified in accordance with the table. When thermocouples are used there are various options for combining the measurement type (coding A) with the measuring range (coding B). In this case, the low word of parameter `Mode` must be calculated according to the following formula and the result entered at the Mode input parameter as an INTEGER value:

Measuring range coding = 256 · Code A + Code B

Please note: The table shows codes **A** and **B** in binary format, and the measuring range coding as a hexadecimal number as the result.

| Measuring type | Code (A) | Measuring range | Code(B) | Measuring range coding (256*A+B) |
|---|---|---|---|---|
| Disabled | | | | 16#0000 |
| Voltage | 2#0001 | ± 25 mV | 2#1010 | 16#010A |
| | | ± 50mV | 2#1011 | 16#010B |
| | | ± 80 mV | 2#0001 | 16#0101 |
| | | ± 250 mV | 2#0010 | 16#0102 |
| | | ± 500 mV | 2#0011 | 16#0103 |
| | | ± 1 V | 2#0100 | 16#0104 |
| | | ± 2,5 V | 2#0101 | 16#0105 |
| | | ± 5 V | 2#0110 | 16#0106 |
| | | 1 to 5 V | 2#0111 | 16#0107 |
| | | 0 to 10V | 2#1000 | 16#0108 |
| | | ± 10 V | 2#1001 | 16#0109 |
| | | ± 100 mV | 2#1100 | 16#010C |
| 4-wire measuring transducer | 2#0010 | ± 3.2 mA | 2#0000 | 16#0200 |
| | | ± 5 mA | 2#0101 | 16#0205 |
| | | ± 10 mA | 2#0001 | 16#0201 |
| | | 0 to 20 mA | 2#0010 | 16#0202 |
| | | 4 to 20 mA | 2#0011 | 16#0203 |
| | | ± 20 mA | 2#0100 | 16#0204 |
| HART interface | 2#0111 | 4 to 20 mA (variant 0) | 2#1100 | 16#070C |
| 2-wire measuring transducer | 2#0011 | 0 to 20 mA | 2#0010 | 16#0302 |
| | | 4 to 20 mA | 2#0011 | 16#0303 |
| | | ± 20 mA | 2#0100 | 16#0304 |
| | | 0 to 10 mA | 2#0110 | 16#0306 |
| Resistor 4-wire connection | 2#0100 | 48 Ω | 2#0000 | 16#0400 |
| | | 150 Ω | 2#0010 | 16#0402 |
| | | 300 Ω | 2#0100 | 16#0404 |
| | | 600 Ω | 2#0110 | 16#0406 |
| | | 1000 Ω | 2#0111 | 16#040E |
| | | 3000 Ω | 2#0111 | 16#0407 |
| | | 6000 Ω | 2#1000 | 16#0408 |
| | | PTC | 2#1111 | 16#040F |

| Measuring type | Code (A) | Measuring range | Code(B) | Measuring range coding (256*A+B) |
|---|---|---|---|---|
| Resistor 3-wire connection | 2#0101 | 48 Ω | 2#0000 | 16#0500 |
| | | 150 Ω | 2#0010 | 16#0502 |
| | | 300 Ω | 2#0100 | 16#0504 |
| | | 600 Ω | 2#0110 | 16#0506 |
| | | 1000 Ω | 2#0111 | 16#050E |
| | | 3000 Ω | 2#0111 | 16#0507 |
| | | 6000 Ω | 2#1000 | 16#0508 |
| | | PTC | 2#1111 | 16#050F |
| Resistor 2-wire connection | 2#0110 | 48 Ω | 2#0000 | 16#0600 |
| | | 150 Ω | 2#0010 | 16#0602 |
| | | 300 Ω | 2#0100 | 16#0604 |
| | | 600 Ω | 2#0110 | 16#0606 |
| | | 1000 Ω | 2#0111 | 16#060E |
| | | 3000 Ω | 2#0111 | 16#0607 |
| | | 6000 Ω | 2#1000 | 16#0608 |
| | | PTC | 2#1111 | 16#060F |

| Measuring type | Code (A) | Measuring range | Code(B) | Measuring range coding (256*A+B) |
|---|---|---|---|---|
| Thermal resistance, linear, 4-wire connection | 2#1000 | Pt 100 climate range | 2#0000 | 16#0800 |
| | | Pt 200 climate range | 2#0111 | 16#0807 |
| | | Pt 500 climate range | 2#1000 | 16#0808 |
| | | Pt 1000 climate range | 2#1001 | 16#0809 |
| | | Ni 100 climate range | 2#0001 | 16#0801 |
| | | Ni 1000 / LG-Ni 1000 climatic range | 2#1010 | 16#080A |
| | | Pt 100 standard range | 2#0010 | 16#0802 |
| | | Pt 200 standard range | 2#0011 | 16#0803 |
| | | Pt 500 standard range | 2#0100 | 16#0804 |
| | | Pt 1000 standard range | 2#0101 | 16#0805 |
| | | Ni 100 standard range | 2#1011 | 16#080B |
| | | Ni 1000 / LG-Ni 1000 standard range | 2#0110 | 16#0806 |
| | | Ni 120 standard range | 2#1100 | 16#080C |
| | | Ni 120 climate range | 2#1101 | 16#080D |
| | | Cu 10 climate range | 2#1110 | 16#080E |
| | | Cu 10 standard range | 2#1111 | 16#080F |
| | | Ni 200 standard range | 2#10000 | 16#0810 |
| | | Ni 200 climate range | 2#10001 | 16#0811 |
| | | Ni 500 standard range | 2#10010 | 16#0812 |
| | | Ni 500 climate range | 2#10011 | 16#0813 |
| | | Pt 10 GOST climatic | 2#10100 | 16#0814 |
| | | Pt 10 GOST standard (TC = 3910) | 2#10101 | 16#0815 |
| | | Pt 50 GOST climatic | 2#10110 | 16#0816 |
| | | Pt 50 GOST standard (TC = 3910) | 2#10111 | 16#0817 |
| | | Pt 100 GOST climatic | 2#11000 | 16#0818 |
| | | Pt 100 GOST standard (TC = 3910) | 2#11001 | 16#0819 |
| | | Pt 500 GOST climatic | 2#11010 | 16#081A |
| | | Pt 500 GOST standard (TC = 3910) | 2#11011 | 16#081B |
| | | Cu 10 GOST climatic | 2#11100 | 16#081C |
| | | Cu 10 GOST standard (TC = 426) | 2#11101 | 16#081D |
| | | Cu 50 GOST climatic | 2#11110 | 16#081E |
| | | Cu 50 GOST standard (TC = 426) | 2#11111 | 16#081F |
| | | Cu 100 GOST climatic | 2#100000 | 16#0820 |
| | | Cu 100 GOST standard (TC = 426) | 2#100001 | 16#0821 |
| | | Ni 100 GOST climatic | 2#100010 | 16#0822 |

| Measuring type | Code (A) | Measuring range | Code(B) | Measuring range coding (256*A+B) |
|---|---|---|---|---|
| | | Ni 100 GOST standard | 2#100011 | 16#0823 |
| | | Pt 10 GOST standard (TC = 3850) | 2#101010 1 | 16#0855 |
| | | Pt 50 GOST standard (TC = 3850) | 2#101011 1 | 16#0857 |
| | | Pt 100 GOST standard (TC = 3850) | 2#101100 1 | 16#0859 |
| | | Pt 500 GOST standard (TC = 3850) | 2#101101 1 | 16#085B |
| | | Cu 10 GOST standard (TC= 428) | 2#100111 01 | 16#089D |
| | | Cu 50 GOST standard (TC= 428) | 2#100111 11 | 16#089F |
| | | Cu 100 GOST standard (TC= 428) | 2#101000 01 | 16#08A1 |

| Measuring type | Code (A) | Measuring range | Code(B) | Measuring range coding (256*A+B) |
|---|---|---|---|---|
| Thermal resistance, linear, 3-wire connection | 2#1001 | Pt 100 climate range | 2#0000 | 16#0900 |
| | | Pt 200 climate range | 2#0111 | 16#0907 |
| | | Pt 500 climate range | 2#1000 | 16#0908 |
| | | Pt 1000 climate range | 2#1001 | 16#0909 |
| | | Ni 100 climate range | 2#0001 | 16#0901 |
| | | Ni 1000 / LG-Ni 1000 climatic range | 2#1010 | 16#090A |
| | | Pt 100 standard range | 2#0010 | 16#0902 |
| | | Pt 200 standard range | 2#0011 | 16#0903 |
| | | Pt 500 standard range | 2#0100 | 16#0904 |
| | | Pt 1000 standard range | 2#0101 | 16#0905 |
| | | Ni 100 standard range | 2#1011 | 16#090B |
| | | Ni 1000 / LG-Ni 1000 standard range | 2#0110 | 16#0906 |
| | | Ni 120 standard range (variant 0)<br>KTY83/110 (variant 1) | 2#1100 | 16#090C |
| | | Ni 120 climate range (variant 0)<br>KTY84/130 (variant 1) | 2#1101 | 16#090D |
| | | Cu 10 climate range | 2#1110 | 16#090E |
| | | Cu 10 standard range | 2#1111 | 16#090F |
| | | Ni 200 standard range | 2#10000 | 16#0910 |
| | | Ni 200 climate range | 2#10001 | 16#0911 |
| | | Ni 500 standard range | 2#10010 | 16#0912 |
| | | Ni 500 climate range | 2#10011 | 16#0913 |
| | | Pt 10 GOST climatic | 2#10100 | 16#0914 |
| | | Pt 10 GOST standard (TC = 3910) | 2#10101 | 16#0915 |
| | | Pt 50 GOST climatic | 2#10110 | 16#0916 |
| | | Pt 50 GOST standard (TC = 3910) | 2#10111 | 16#0917 |
| | | Pt 100 GOST climatic | 2#11000 | 16#0918 |
| | | Pt 100 GOST standard (TC = 3910) | 2#11001 | 16#0919 |
| | | Pt 500 GOST climatic | 2#11010 | 16#091A |
| | | Pt 500 GOST standard (TC = 3910) | 2#11011 | 16#091B |
| | | Cu 10 GOST climatic | 2#11100 | 16#091C |
| | | Cu 10 GOST standard (TC = 426) | 2#11101 | 16#091D |
| | | Cu 50 GOST climatic | 2#11110 | 16#091E |
| | | Cu 50 GOST standard (TC = 426) | 2#11111 | 16#091F |

| Measuring type | Code (A) | Measuring range | Code(B) | Measuring range coding (256*A+B) |
|---|---|---|---|---|
| | | Cu 100 GOST climatic | 2#100000 | 16#0920 |
| | | Cu 100 GOST standard (TC = 426) | 2#100001 | 16#0921 |
| | | Ni 100 GOST climatic | 2#100010 | 16#0922 |
| | | Ni 100 GOST standard | 2#100011 | 16#0923 |
| | | Pt 10 GOST standard (TC = 3850) | 2#1010101 | 16#0955 |
| | | Pt 50 GOST standard (TC = 3850) | 2#1010111 | 16#0957 |
| | | Pt 100 GOST standard (TC = 3850) | 2#1011001 | 16#0959 |
| | | Pt 500 GOST standard (TC = 3850) | 2#1011011 | 16#095B |
| | | Cu 10 GOST standard (TC= 428) | 2#10011101 | 16#099D |
| | | Cu 50 GOST standard (TC= 428) | 2#10011111 | 16#099F |
| | | Cu 100 GOST standard (TC= 428) | 2#10100001 | 16#09A1 |
| Thermal resistance, linear, 2-wire connection | 2#1111 | Pt 100 climate range | 2#0000 | 16#0F00 |
| | | Pt 200 climate range | 2#0111 | 16#0F07 |
| | | Pt 500 climate range | 2#1000 | 16#0F08 |
| | | Pt 1000 climate range | 2#1001 | 16#0F09 |
| | | Ni 100 climate range | 2#0001 | 16#0F01 |
| | | Ni 1000 / LG-Ni 1000 climatic range | 2#1010 | 16#0F0A |
| | | Pt 100 standard range | 2#0010 | 16#0F02 |
| | | Pt 200 standard range | 2#0011 | 16#0F03 |
| | | Pt 500 standard range | 2#0100 | 16#0F04 |
| | | Pt 1000 standard range | 2#0101 | 16#0F05 |
| | | Ni 100 standard range | 2#1011 | 16#0F0B |
| | | Ni 1000 / LG-Ni 1000 standard range | 2#0110 | 16#0F06 |
| | | Ni 120 standard range | 2#1100 | 16#0F0C |
| | | Ni 120 climate range | 2#1101 | 16#0F0D |
| | | Cu 10 climate range | 2#1110 | 16#0F0E |
| | | Cu 10 standard range | 2#1111 | 16#0F0F |
| | | Ni 200 standard range | 2#10000 | 16#0F10 |
| | | Ni 200 climate range | 2#10001 | 16#0F11 |
| | | Ni 500 standard range | 2#10010 | 16#0F12 |
| | | Ni 500 climate range | 2#10011 | 16#0F13 |

| Measuring type | Code (A) | Measuring range | Code(B) | Measuring range coding (256*A+B) |
|---|---|---|---|---|
| Thermocouple, linear, reference temperature 0 °C / No reference point | 2#1010 | Type B [PtRh-PtRh] | 2#0000 | 16#0A00 |
| | | Type N [NiCrSi-NiSi] | 2#0001 | 16#0A01 |
| | | Type E [NiCr-CuNi] | 2#0010 | 16#0A02 |
| | | Type R [PtRh-Pt] | 2#0011 | 16#0A03 |
| | | Type S [PtRh-Pt] | 2#0100 | 16#0A04 |
| | | Type J [Fe-CuNi IEC] | 2#0101 | 16#0A05 |
| | | Type L [Fe-CuNi DIN] | 2#0110 | 16#0A06 |
| | | Type T [Cu-CuNi IEC] | 2#0111 | 16#0A07 |
| | | Type K [NiCr-Ni] | 2#1000 | 16#0A08 |
| | | Type U [Cu-CuNi DIN] | 2#1001 | 16#0A09 |
| | | Type C | 2#1010 | 16#0A0A |
| | | Type TXK/XK(L) | 2#1011 | 16#0A0B |
| Thermocouple, linear, reference temperature 50 °C | 2#1011 | Type B [PtRh-PtRh] | 2#0000 | 16#0B00 |
| | | Type N [NiCrSi-NiSi] | 2#0001 | 16#0B01 |
| | | Type E [NiCr-CuNi] | 2#0010 | 16#0B02 |
| | | Type R [PtRh-Pt] | 2#0011 | 16#0B03 |
| | | Type S [PtRh-Pt] | 2#0100 | 16#0B04 |
| | | Type J [Fe-CuNi IEC] | 2#0101 | 16#0B05 |
| | | Type L [Fe-CuNi DIN] | 2#0110 | 16#0B06 |
| | | Type T [Cu-CuNi IEC] | 2#0111 | 16#0B07 |
| | | Type K [NiCr-Ni] | 2#1000 | 16#0B08 |
| | | Type U [Cu-CuNi DIN] | 2#1001 | 16#0B09 |
| | | Type C | 2#1010 | 16#0B0A |
| | | Type TXK/XK(L) | 2#1011 | 16#0B0B |
| Thermocouple, fixed ref. temp | 2#1100 | Type B [PtRh-PtRh] | 2#0000 | 16#0C00 |
| | | Type N [NiCrSi-NiSi] | 2#0001 | 16#0C01 |
| | | Type E [NiCr-CuNi] | 2#0010 | 16#0C02 |
| | | Type R [PtRh-Pt] | 2#0011 | 16#0C03 |
| | | Type S [PtRh-Pt] | 2#0100 | 16#0C04 |
| | | Type J [Fe-CuNi IEC] | 2#0101 | 16#0C05 |
| | | Type L [Fe-CuNi DIN] | 2#0110 | 16#0C06 |
| | | Type T [Cu-CuNi IEC] | 2#0111 | 16#0C07 |
| | | Type K [NiCr-Ni] | 2#1000 | 16#0C08 |
| | | Type U [Cu-CuNi DIN] | 2#1001 | 16#0C09 |
| | | Type C | 2#1010 | 16#0C0A |
| | | Type TXK/XK(L) | 2#1011 | 16#0C0B |

| Measuring type | Code (A) | Measuring range | Code(B) | Measuring range coding (256*A+B) |
|---|---|---|---|---|
| Thermocouple, linear, internal compensation/ internal reference point | 2#1101 | Type B [PtRh-PtRh] | 2#0000 | 16#0D00 |
| | | Type N [NiCrSi-NiSi] | 2#0001 | 16#0D01 |
| | | Type E [NiCr-CuNi] | 2#0010 | 16#0D02 |
| | | Type R [PtRh-Pt] | 2#0011 | 16#0D03 |
| | | Type S [PtRh-Pt] | 2#0100 | 16#0D04 |
| | | Type J [Fe-CuNi IEC] | 2#0101 | 16#0D05 |
| | | Type L [Fe-CuNi DIN] | 2#0110 | 16#0D06 |
| | | Type T [Cu-CuNi IEC] | 2#0111 | 16#0D07 |
| | | Type K [NiCr-Ni] | 2#1000 | 16#0D08 |
| | | Type U [Cu-CuNi DIN] | 2#1001 | 16#0D09 |
| | | Type C | 2#1010 | 16#0D0A |
| | | Type TXK/XK(L) | 2#1011 | 16#0D0B |
| Thermocouple, linear, internal compensation/ reference point RTD (0) | 2#1110 | Type B [PtRh-PtRh] | 2#0000 | 16#0E00 |
| | | Type N [NiCrSi-NiSi] | 2#0001 | 16#0E01 |
| | | Type E [NiCr-CuNi] | 2#0010 | 16#0E02 |
| | | Type R [PtRh-Pt] | 2#0011 | 16#0E03 |
| | | Type S [PtRh-Pt] | 2#0100 | 16#0E04 |
| | | Type J [Fe-CuNi IEC] | 2#0101 | 16#0E05 |
| | | Type L [Fe-CuNi DIN] | 2#0110 | 16#0E06 |
| | | Type T [Cu-CuNi IEC] | 2#0111 | 16#0E07 |
| | | Type K [NiCr-Ni] | 2#1000 | 16#0E08 |
| | | Type U [Cu-CuNi DIN] | 2#1001 | 16#0E09 |
| | | Type C | 2#1010 | 16#0E0A |
| | | Type TXK/XK(L) | 2#1011 | 16#0E0B |
| Thermocouple, dynamic ref. temp | 2#0111 | Type B [PtRh-PtRh] (variant 1) | 2#0000 | 16#0700 |
| | | Type N [NiCrSi-NiSi] (variant 1) | 2#0001 | 16#0701 |
| | | Type E [NiCr-CuNi] (variant 1) | 2#0010 | 16#0702 |
| | | Type R [PtRh-Pt] (variant 1) | 2#0011 | 16#0703 |
| | | Type S [PtRh-Pt] (variant 1) | 2#0100 | 16#0704 |
| | | Type J [Fe-CuNi IEC] (variant 1) | 2#0101 | 16#0705 |
| | | Type L [Fe-CuNi DIN] (variant 1) | 2#0110 | 16#0706 |
| | | Type T [Cu-CuNi IEC] (variant 1) | 2#0111 | 16#0707 |
| | | Type K [NiCr-Ni] (variant 1) | 2#1000 | 16#0708 |

### Effect of the temperature coefficients on the measuring range

- Setting TC = 3850 at GOST Standard Pt 10, Pt 50, Pt 100, Pt 500 sets Bit 7 within the measuring range byte (0 x 40)
- Setting TC = 428 at GOST Standard Cu 10, Cu 50, Cu 100 sets Bit 8 within the measuring range byte (0 x 80)

### Measuring range coding of the analog output modules

Depending on the measuring range coding of the analog output modules, the parameter `Mode` (measuring-range coding) corresponding to the channel must be specified in accordance with the table.

| Measuring type | Measuring range | Mode |
|---|---|---|
| Voltage | ± 5 V | 16#0106 |
| | 1 to 5 V | 16#0107 |
| | 0 to 10V | 16#0108 |
| | ± 10 V | 16#0109 |
| Current | 0 to 10 mA | 16#0206 |
| | 0 to 20 mA | 16#0202 |
| | 4 to 20 mA | 16#0203 |
| | ± 20 mA | 16#0204 |
| HART interface | 4 to 20 mA | 16#070C |

### Measuring-Range Coding of the Digital Input and Output Modules

There is no measuring type and no measuring range for digital input modules and digital output modules:

- `Mode` **= 16#FFFF** (with DI)
- `Mode` **= 16#FFFE** (with DO)

### Measuring range coding of the controller module

There is no measuring type and no measuring range for controller modules:

- `Mode` **= 16#FFFD**

### Measuring range coding of the ET 200SP Analog Input Energy Meter module

There is no measuring type and no measuring range for analoge input energy meter moduls:

- Mode = **16#200C**

## 17.24.2 Mode settings for field devices

### Mode structure

The structure and meaning of the input Mode of data type DWORD are shown below:

| | | |
|---|---|---|
| Byte 3: | 16#80: Value status "valid value"<br>16#00: Value status "invalid value"<br>16#40: Value status "invalid value" | <br>(Channel error)<br>(Higher-level error) |
| Byte 2: | 16#01: Restart (OB 100) has been carried out<br>16#xy: Variant identification with multiple mode assignment (see below) | |
| Byte 1, 0 (low word): | Measuring range coding (see below) | |

Example:

16#80010001 = Value status "valid value", restart has been carried out, PA device with analog input

### PA/FF device variant identifier

| Variant | x | y | Field device |
|---|---|---|---|
| 0 | 0 | 0 | PA device |
| 1 | 1 | 0 | FF device |

### Mode settings for field devices

MODE_xx input parameters are available for a maximum of 32 slots of a PA/FF field device. Their initial value is zero (no read/write access). You must set the combination selected from the options of the PROFIBUS PA 3.0 profile at the MODE_xx input of each slot channel xx:

| Block | Parameter at block ... | ...and at field device | Input/output from block view (PLS view) | Mode 16#xxyy<br>O = xx<br>I = yy | Configuration possible with field device |
|---|---|---|---|---|---|
| Analog input (PA_AI/ FbAnIn) | PV | OUT | I | 16#0001 | PA + FF |
| Totalizer (FbAnTot) | PV | | I | 16#000F | PA + FF |
| Totalizer (FbAnTot) | PV<br>PV_Set | | I<br>O | 16#070F | PA + FF |
| Totalizer (FbAnTot) | PV<br>PV_Set<br>PV_Mode | | I<br>O<br>O | 16#080F | PA + FF |
| Analog output (FbAnOu) | SP | SP | O | 16#0100 | PA + FF |

| Block | Parameter at block ... | ...and at field device | Input/output from block view (PLS view) | Mode 16#xxyy<br>O = xx<br>I = yy | Configuration possible with field device |
|---|---|---|---|---|---|
| Analog output (FbAnOu) | SP<br>Rbk<br>PosD | SP<br>READBACK<br>POS_D | O<br>I<br>I | 16#0103 | PA |
| Analog output (FbAnOu) | SP<br>CbkBy0 - CbkBy2 | SP<br>CHECK_BACK | O<br>I | 16#0104 | PA |
| Analog output (FbAnOu) | SP<br>Rbk<br>PosD<br>CbkBy0 - CbkBy2 | SP<br>READBACK<br>POS_D<br>CHECK_BACK | O<br>I<br>I<br>I | 16#0105 | PA |
| Analog output (FbAnOu) | RCasIn,<br>RCasOut | RCAS_IN<br>RCAS_OUT | O<br>I | 16#0206 | PA |
| Analog output (FbAnOu) | RCasIn<br>RCasOut<br>CbkBy0 - CbkBy2 | RCAS_IN<br>RCAS_OUT<br>CHECK_BACK | O<br>I<br>I | 16#0207 | PA |
| Analog output (FbAnOu) | SP<br>RCasIn<br>Rbk<br>RCasOut<br>PosD<br>CbkBy0 - CbkBy2 | SP<br>RCAS_IN<br>READBACK<br>RCAS_OUT<br>POS_D<br>CHECK_BACK | O<br>O<br>I<br>I<br>I<br>I | 16#0308 | PA |
| Binary input (FbDiIn) | PV | OUT_D | I | 16#0002 | PA + FF |
| Binary output (FbDiOu ) | SP | SP_D | O | 16#0400 | PA + FF |
| Binary output (FbDiOu ) | SP<br>Rbk | SP_D<br>READBACK_D | O<br>I | 16#0409 | PA |
| Binary output (FbDiOu) | SP<br>CbkBy0 - CbkBy2 | SP_D<br>CHECKBACK_D | O<br>I | 16#040A | PA |
| Binary output (FbDiOu) | SP<br>Rbk<br>CbkBy0 - CbkBy2 | SP_D<br>READBACK_D<br>CHECK_BACK_D | O<br>I<br>I | 16#040B | PA |
| Binary output (FbDiOu) | RCasIn<br>RCasOut | RCAS_IN_D<br>RCAS_OUT_D | O<br>I | 16#050C | PA |
| Binary output (FbDiOu) | RCasIn<br>RCasOut<br>CbkBy0 - CbkBy2 | RCAS_IN_D<br>RCAS_OUT_D<br>CHECK_BACK_D | O<br>I<br>I | 16#050D | PA |
| Binary output (FbDiOu) | SP<br>RCasIn<br>Rbk<br>RCasOut<br>CbkBy0 - CbkBy2 | SP_D<br>RCAS_IN_D<br>READBACK_D<br>RCAS_OUT_D<br>CHECK_BACK_D | O<br>O<br>I<br>I<br>I | 16#060E | PA |

## 17.24.3 Mode settings for Channel blocks

### Mode Structure

Analog input, byte1=16#01 (voltage):

| Measuring type | Measuring range | MODE |
|---|---|---|
| Voltage | 0V to 1V | 16#010D |
| | -20V to 0V | 16#010E |
| | 0V to 20V | 16#0110 |

Analog input, byte1=16#02 (current / 4wire):

| Measuring type | Measuring range | MODE |
|---|---|---|
| Current | -20mA to 0mA | 16#0207 |
| | +/-30mA | 16#0208 |

Analog input, byte1=16#03 (current / 2wire):

| Measuring type | Measuring range | MODE |
|---|---|---|
| Current | -20mA to 0mA | 16#0307 |
| | +/-30mA | 16#0308 |

# Conversion blocks 18

## 18.1 StrgToBy - String in byte structure (Struct of Byte)

### 18.1.1 Description of StrgToBy

#### Object name (type + number) and family

Type + number: FC 384

Family: Convert

#### Area of application for StrgToBy

The block converts a string with a maximum length of 32 into a byte structure (Struct of Byte). This enables strings to be interconnected via the StrgToBy block to the inputs for the associated values (`ExtValxx`) of the technological blocks.

#### How it works

The block uses the input string "`StringIn`", converts it to a byte structure and transfers it to the output parameter `StrctOut`".

#### Configuration

Use the CFC editor to install the block in any OB.

#### Startup characteristics

The block does not have any startup characteristics.

#### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

#### See also

StrgToBy modes (Page 2702)

StrgToBy functions (Page 2702)

StrgToBy error handling (Page 2703)

StrgToBy messaging (Page 2703)

StrgToBy I/Os (Page 2703)
StrgToBy block diagram (Page 2704)

## 18.1.2 StrgToBy modes

### Operating modes of `StrgToBy`

This block does not have any operating modes.

### See also

Description of StrgToBy (Page 2701)
StrgToBy functions (Page 2702)
StrgToBy error handling (Page 2703)
StrgToBy messaging (Page 2703)
StrgToBy I/Os (Page 2703)
StrgToBy block diagram (Page 2704)

## 18.1.3 StrgToBy functions

### Functions of StrgToBy

The block uses the input string "`StringIn`", converts it to a byte structure and transfers it to the output parameter `StrctOut`".

### See also

Description of StrgToBy (Page 2701)
StrgToBy modes (Page 2702)
StrgToBy error handling (Page 2703)
StrgToBy messaging (Page 2703)
StrgToBy I/Os (Page 2703)
StrgToBy block diagram (Page 2704)

## 18.1.4 StrgToBy error handling

### Error handling of StrgToBy

The block does not report any errors.

### See also

Description of StrgToBy (Page 2701)
StrgToBy modes (Page 2702)
StrgToBy functions (Page 2702)
StrgToBy messaging (Page 2703)
StrgToBy I/Os (Page 2703)
StrgToBy block diagram (Page 2704)

## 18.1.5 StrgToBy messaging

### Messaging

This block does not offer messaging.

### See also

Description of StrgToBy (Page 2701)
StrgToBy modes (Page 2702)
StrgToBy functions (Page 2702)
StrgToBy error handling (Page 2703)
StrgToBy I/Os (Page 2703)
StrgToBy block diagram (Page 2704)

## 18.1.6 StrgToBy I/Os

I/Os of `StrgToBy`

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| StringIn | Input string | STRING[32] | '' |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 1 |
| StrctOut | Byte structure | STRUCT<br>• Byte1:BYTE<br>• Byte2:BYTE<br>• Byte3:BYTE<br>• ...<br>• Byte32:BYTE | -<br>• 16#00<br>• 16#00<br>• 16#00<br>• 16#00<br>• 16#00 |

### See also

Description of StrgToBy (Page 2701)
StrgToBy modes (Page 2702)
StrgToBy functions (Page 2702)
StrgToBy error handling (Page 2703)
StrgToBy messaging (Page 2703)
StrgToBy block diagram (Page 2704)

## 18.1.7 StrgToBy block diagram

### Block diagram of StrgToBy

A block diagram is not provided for this block.

### See also

Description of StrgToBy (Page 2701)
StrgToBy modes (Page 2702)
StrgToBy functions (Page 2702)
StrgToBy error handling (Page 2703)

StrgToBy messaging (Page 2703)
StrgToBy I/Os (Page 2703)

# 18.2 StruAnIn - Separating an analog structured variable

## 18.2.1 Description of StruAnIn

### Object name (type + number) and family

Type + number: FC 375

Family: Convert

### Area of application for StruAnIn

The block is used for the following applications:

- Separation of an analog value with a structure into a variable of the REAL data type and a signal status

### How it works

The block separates an analog value with a structure interconnected to the `In` input parameter into a variable (`Out`) of the REAL data type and a (`ST`) signal status.

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for parameter `Status`

This block does not have the `Status` parameter.

### See also

StruAnIn block diagram (Page 2709)

StruAnIn I/Os (Page 2708)

StruAnIn messaging (Page 2708)

StruAnIn error handling (Page 2707)

StruAnIn functions (Page 2707)

StruAnIn modes (Page 2707)

## 18.2.2 StruAnIn modes

### StruAnIn modes

This block does not have any modes.

### See also

StruAnIn block diagram (Page 2709)
StruAnIn I/Os (Page 2708)
StruAnIn messaging (Page 2708)
StruAnIn error handling (Page 2707)
StruAnIn functions (Page 2707)
Description of StruAnIn (Page 2706)

## 18.2.3 StruAnIn functions

### Functions of StruAnIn

There are no other functions for this block.

### See also

StruAnIn block diagram (Page 2709)
StruAnIn I/Os (Page 2708)
StruAnIn messaging (Page 2708)
StruAnIn error handling (Page 2707)
StruAnIn modes (Page 2707)
Description of StruAnIn (Page 2706)

## 18.2.4 StruAnIn error handling

### StruAnIn error handling

The block does not report any errors.

### See also

StruAnIn block diagram (Page 2709)
StruAnIn I/Os (Page 2708)
StruAnIn messaging (Page 2708)
StruAnIn functions (Page 2707)
StruAnIn modes (Page 2707)
Description of StruAnIn (Page 2706)

## 18.2.5 StruAnIn messaging

### Messaging

This block does not offer messaging.

### See also

StruAnIn block diagram (Page 2709)
StruAnIn I/Os (Page 2708)
StruAnIn error handling (Page 2707)
StruAnIn functions (Page 2707)
StruAnIn modes (Page 2707)
Description of StruAnIn (Page 2706)

## 18.2.6 StruAnIn I/Os

### StruAnIn I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In | Analog value with structure | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ST | Signal status | BYTE | 16#80 |
| Value | Analog value | REAL | 0.0 |

### See also

StruAnIn block diagram (Page 2709)
StruAnIn messaging (Page 2708)
StruAnIn error handling (Page 2707)
StruAnIn functions (Page 2707)
StruAnIn modes (Page 2707)
Description of StruAnIn (Page 2706)

## 18.2.7 StruAnIn block diagram

### StruAnIn block diagram

A block diagram is not provided for this block.

### See also

StruAnIn I/Os (Page 2708)
StruAnIn messaging (Page 2708)
StruAnIn error handling (Page 2707)
StruAnIn functions (Page 2707)
StruAnIn modes (Page 2707)
Description of StruAnIn (Page 2706)

# 18.3 StruAnOu - Creating an analog structured variable

## 18.3.1 Description of StruAnOu

### Object name (type + number) and family

Type + number: FC 376

Family: Convert

### Area of application for StruAnOu

The block is used for the following applications:

- Merging a variable of the REAL data type and a signal status into an analog process value.

### How it works

The block merges an analog value (`Value`) of the REAL data type and a signal status (`ST`) to form an analog value (`Out`) with a structure.

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

StruAnOu modes (Page 2711)
StruAnOu functions (Page 2711)
StruAnOu error handling (Page 2711)
StruAnOu messaging (Page 2712)
StruAnOu I/Os (Page 2712)
StruAnOu block diagram (Page 2713)

## 18.3.2 StruAnOu modes

### StruAnOu modes

This block does not have any modes.

### See also

Description of StruAnOu (Page 2710)
StruAnOu functions (Page 2711)
StruAnOu error handling (Page 2711)
StruAnOu messaging (Page 2712)
StruAnOu I/Os (Page 2712)
StruAnOu block diagram (Page 2713)

## 18.3.3 StruAnOu functions

### Functions of StruAnOu

There are no other functions for this block.

### See also

Description of StruAnOu (Page 2710)
StruAnOu modes (Page 2711)
StruAnOu error handling (Page 2711)
StruAnOu messaging (Page 2712)
StruAnOu I/Os (Page 2712)
StruAnOu block diagram (Page 2713)

## 18.3.4 StruAnOu error handling

### StruAnOu error handling

The block does not report any errors.

### See also

Description of StruAnOu (Page 2710)
StruAnOu modes (Page 2711)
StruAnOu functions (Page 2711)
StruAnOu messaging (Page 2712)
StruAnOu I/Os (Page 2712)
StruAnOu block diagram (Page 2713)

## 18.3.5 StruAnOu messaging

### Messaging

This block does not offer messaging.

### See also

Description of StruAnOu (Page 2710)
StruAnOu modes (Page 2711)
StruAnOu functions (Page 2711)
StruAnOu error handling (Page 2711)
StruAnOu I/Os (Page 2712)
StruAnOu block diagram (Page 2713)

## 18.3.6 StruAnOu I/Os

### StruAnOu I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| ST | Signal status | BYTE | 16#80 |
| Value | Analog value | REAL | 0.0 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Bad | 1 = (ST = 16#00 to 16#3F) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Analog value with structure | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### See also

Description of StruAnOu (Page 2710)
StruAnOu modes (Page 2711)
StruAnOu functions (Page 2711)
StruAnOu error handling (Page 2711)
StruAnOu messaging (Page 2712)
StruAnOu block diagram (Page 2713)

## 18.3.7 StruAnOu block diagram

### StruAnOu block diagram

A block diagram is not provided for this block.

### See also

Description of StruAnOu (Page 2710)
StruAnOu modes (Page 2711)
StruAnOu functions (Page 2711)
StruAnOu error handling (Page 2711)
StruAnOu messaging (Page 2712)
StruAnOu I/Os (Page 2712)

# 18.4 StruDiIn - Separating a digital structured variable

## 18.4.1 Description of StruDiIn

### Object name (type + number) and family

Type + number: FC 377

Family: Convert

### Area of application for StruDiIn

The block is used for the following applications:

- Separating a binary process value into a variable of the BOOL data type, a process value and signal status.

### How it works

The block separates a binary process value interconnected to the `In` input parameter into a variable of the BOOL data type and a signal status.

### Configuration

Use the CFC editor to install the block in any OB

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

StruDiIn block diagram (Page 2717)

StruDiIn I/Os (Page 2716)

StruDiIn messaging (Page 2716)

StruDiIn error handling (Page 2715)

StruDiIn functions (Page 2715)

StruDiIn modes (Page 2715)

## 18.4.2 StruDiIn modes

### StruDiIn modes

This block does not have any modes.

### See also

StruDiIn block diagram (Page 2717)
StruDiIn I/Os (Page 2716)
StruDiIn messaging (Page 2716)
StruDiIn error handling (Page 2715)
StruDiIn functions (Page 2715)
Description of StruDiIn (Page 2714)

## 18.4.3 StruDiIn functions

### Functions of StruDiIn

There are no other functions for this block.

### See also

StruDiIn block diagram (Page 2717)
StruDiIn I/Os (Page 2716)
StruDiIn messaging (Page 2716)
StruDiIn error handling (Page 2715)
StruDiIn modes (Page 2715)
Description of StruDiIn (Page 2714)

## 18.4.4 StruDiIn error handling

### StruDiIn error handling

The block does not report any errors.

### See also

StruDiIn block diagram (Page 2717)
StruDiIn I/Os (Page 2716)
StruDiIn messaging (Page 2716)
StruDiIn functions (Page 2715)
StruDiIn modes (Page 2715)
Description of StruDiIn (Page 2714)

## 18.4.5 StruDiIn messaging

### Messaging

This block does not offer messaging.

### See also

StruDiIn block diagram (Page 2717)
StruDiIn I/Os (Page 2716)
StruDiIn error handling (Page 2715)
StruDiIn functions (Page 2715)
StruDiIn modes (Page 2715)
Description of StruDiIn (Page 2714)

## 18.4.6 StruDiIn I/Os

### StruDiIn I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In | Binary process value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ST | Signal status | BYTE | 16#80 |
| Value | Binary variable | BOOL | 0 |

### See also

StruDiIn block diagram (Page 2717)
StruDiIn messaging (Page 2716)
StruDiIn error handling (Page 2715)
StruDiIn functions (Page 2715)
StruDiIn modes (Page 2715)
Description of StruDiIn (Page 2714)

## 18.4.7 StruDiIn block diagram

### StruDiIn block diagram

A block diagram is not provided for this block.

### See also

StruDiIn I/Os (Page 2716)
StruDiIn messaging (Page 2716)
StruDiIn error handling (Page 2715)
StruDiIn functions (Page 2715)
StruDiIn modes (Page 2715)
Description of StruDiIn (Page 2714)

# 18.5 StruDiOu - Creating a digital structured variable

## 18.5.1 Description of StruDiOu

### Object name (type + number) and family

Type + number: FC 378

Family: Convert

### Area of application for StruDiOu

The block is used for the following applications:

- Merging a variable of the BOOL data type and a signal status into a binary process value.

### How it works

The block merges a variable of the BOOL data type and a signal status into a binary process value.

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

StruDiOu block diagram (Page 2721)

StruDiOu I/Os (Page 2720)

StruDiOu messaging (Page 2720)

StruDiOu error handling (Page 2719)

StruDiOu functions (Page 2719)

StruDiOu modes (Page 2719)

## 18.5.2 StruDiOu modes

### StruDiOu modes

This block does not have any modes.

### See also

StruDiOu block diagram (Page 2721)
StruDiOu I/Os (Page 2720)
StruDiOu messaging (Page 2720)
StruDiOu error handling (Page 2719)
StruDiOu functions (Page 2719)
Description of StruDiOu (Page 2718)

## 18.5.3 StruDiOu functions

### Functions of StruDiOu

There are no other functions for this block.

### See also

StruDiOu block diagram (Page 2721)
StruDiOu I/Os (Page 2720)
StruDiOu messaging (Page 2720)
StruDiOu error handling (Page 2719)
StruDiOu modes (Page 2719)
Description of StruDiOu (Page 2718)

## 18.5.4 StruDiOu error handling

### StruDiOu error handling

The block does not report any errors.

### See also

StruDiOu block diagram (Page 2721)
StruDiOu I/Os (Page 2720)
StruDiOu messaging (Page 2720)
StruDiOu functions (Page 2719)
StruDiOu modes (Page 2719)
Description of StruDiOu (Page 2718)

## 18.5.5 StruDiOu messaging

### Messaging

This block does not offer messaging.

### See also

StruDiOu block diagram (Page 2721)
StruDiOu I/Os (Page 2720)
StruDiOu error handling (Page 2719)
StruDiOu functions (Page 2719)
StruDiOu modes (Page 2719)
Description of StruDiOu (Page 2718)

## 18.5.6 StruDiOu I/Os

### StruDiOu I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| ST | Signal status | BYTE | 16#80 |
| Value | Binary variable | BOOL | 0 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Bad | 1 = (ST = 16#00 to 16#3F) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Binary process value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

### See also

StruDiOu block diagram (Page 2721)
StruDiOu messaging (Page 2720)
StruDiOu error handling (Page 2719)
StruDiOu functions (Page 2719)
StruDiOu modes (Page 2719)
Description of StruDiOu (Page 2718)

## 18.5.7 StruDiOu block diagram

### StruDiOu block diagram

A block diagram is not provided for this block.

### See also

StruDiOu I/Os (Page 2720)
StruDiOu messaging (Page 2720)
StruDiOu error handling (Page 2719)
StruDiOu functions (Page 2719)
StruDiOu modes (Page 2719)
Description of StruDiOu (Page 2718)

# 18.6 StruScIn - Separating a display area into two variables

## 18.6.1 Description of StruScIn

### Object name (type + number) and family

Type + number: FC 379

Family: Convert

### Area of application for StruScIn

The block is used for the following applications:

- Separation of a display area into two variables of data type REAL.

### How it works

The block separates a display area interconnected to the `Scale` input parameter into two variables of the REAL. data type

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

StruScIn block diagram (Page 2725)

StruScIn I/Os (Page 2724)

StruScIn messaging (Page 2724)

StruScIn error handling (Page 2723)

StruScIn functions (Page 2723)

StruScIn modes (Page 2723)

## 18.6.2 StruScIn modes

### StruScIn modes

This block does not have any modes.

### See also

StruScIn block diagram (Page 2725)
StruScIn I/Os (Page 2724)
StruScIn messaging (Page 2724)
StruScIn error handling (Page 2723)
StruScIn functions (Page 2723)
Description of StruScIn (Page 2722)

## 18.6.3 StruScIn functions

### Functions of StruScIn

There are no other functions for this block.

### See also

StruScIn block diagram (Page 2725)
StruScIn I/Os (Page 2724)
StruScIn messaging (Page 2724)
StruScIn error handling (Page 2723)
StruScIn modes (Page 2723)
Description of StruScIn (Page 2722)

## 18.6.4 StruScIn error handling

### StruScIn error handling

The block does not report any errors.

### See also

StruScIn block diagram (Page 2725)
StruScIn I/Os (Page 2724)
StruScIn messaging (Page 2724)
StruScIn functions (Page 2723)
StruScIn modes (Page 2723)
Description of StruScIn (Page 2722)

## 18.6.5 StruScIn messaging

### Messaging

This block does not offer messaging.

### See also

StruScIn block diagram (Page 2725)
StruScIn I/Os (Page 2724)
StruScIn error handling (Page 2723)
StruScIn functions (Page 2723)
StruScIn modes (Page 2723)
Description of StruScIn (Page 2722)

## 18.6.6 StruScIn I/Os

### StruScIn I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `EN` | 1 = Called block will be processed | BOOL | 1 |
| `Scale` | Display area | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| HiScale | High limit of display area | REAL | 100.0 |
| LoScale | Low limit of display area | REAL | 0.0 |

### See also

StruScIn block diagram (Page 2725)
StruScIn messaging (Page 2724)
StruScIn error handling (Page 2723)
StruScIn functions (Page 2723)
StruScIn modes (Page 2723)
Description of StruScIn (Page 2722)

## 18.6.7 StruScIn block diagram

### StruScIn block diagram

A block diagram is not provided for this block.

### See also

StruScIn I/Os (Page 2724)
StruScIn messaging (Page 2724)
StruScIn error handling (Page 2723)
StruScIn functions (Page 2723)
StruScIn modes (Page 2723)
Description of StruScIn (Page 2722)

# 18.7 StruScOu - Merging two variables into a display area

## 18.7.1 Description of StruScOu

### Object name (type + number) and family

Type + number: FC 380

Family: Convert

### Area of application for StruScOu

The block is used for the following applications:

- Merging two variables of the REAL data type into a display area.

### How it works

The block merges two variables of the REAL data type into a display area.

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

StruScOu block diagram (Page 2729)
StruScOu I/Os (Page 2728)
StruScOu messaging (Page 2728)
StruScOu error handling (Page 2727)
StruScOu functions (Page 2727)
StruScOu modes (Page 2727)

## 18.7.2 StruScOu modes

### StruScOu modes

This block does not have any modes.

### See also

StruScOu block diagram (Page 2729)
StruScOu I/Os (Page 2728)
StruScOu messaging (Page 2728)
StruScOu error handling (Page 2727)
StruScOu functions (Page 2727)
Description of StruScOu (Page 2726)

## 18.7.3 StruScOu functions

### Functions of StruScOu

There are no other functions for this block.

### See also

StruScOu block diagram (Page 2729)
StruScOu I/Os (Page 2728)
StruScOu messaging (Page 2728)
StruScOu error handling (Page 2727)
StruScOu modes (Page 2727)
Description of StruScOu (Page 2726)

## 18.7.4 StruScOu error handling

### StruScOu error handling

The block does not report any errors.

### See also

StruScOu block diagram (Page 2729)
StruScOu I/Os (Page 2728)
StruScOu messaging (Page 2728)
StruScOu functions (Page 2727)
StruScOu modes (Page 2727)
Description of StruScOu (Page 2726)

## 18.7.5 StruScOu messaging

### Messaging

This block does not offer messaging.

### See also

StruScOu block diagram (Page 2729)
StruScOu I/Os (Page 2728)
StruScOu error handling (Page 2727)
StruScOu functions (Page 2727)
StruScOu modes (Page 2727)
Description of StruScOu (Page 2726)

## 18.7.6 StruScOu I/Os

### StruScOu I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| HiScale | High limit of display area | REAL | 100.0 |
| LoScale | Low limit of display area | REAL | 0.0 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Scale | Display area | STRUCT<br>• High: REAL<br>• Low: REAL | -<br>• 100.0<br>• 0.0 |

### See also

StruScOu block diagram (Page 2729)
StruScOu messaging (Page 2728)
StruScOu error handling (Page 2727)
StruScOu functions (Page 2727)
StruScOu modes (Page 2727)
Description of StruScOu (Page 2726)

## 18.7.7 StruScOu block diagram

### StruScOu block diagram

A block diagram is not provided for this block.

### See also

StruScOu I/Os (Page 2728)
StruScOu messaging (Page 2728)
StruScOu error handling (Page 2727)
StruScOu functions (Page 2727)
StruScOu modes (Page 2727)
Description of StruScOu (Page 2726)

# 18.8 STIn - Separating the signal status into individual binary displays

## 18.8.1 Description of STIn

### Object name (type + number) and family

Type + number: FC 373

Family: Convert

### Area of application for STIn

The block is used for the following applications:

- Separation of signal status into individual binary displays

### How it works

The block separates a signal status interconnected to the input parameter into individual binary displays.

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

STIn block diagram (Page 2733)

STIn I/Os (Page 2732)

STIn messaging (Page 2732)

STIn error handling (Page 2731)

STIn functions (Page 2731)

STIn modes (Page 2731)

## 18.8.2 STIn modes

### STIn modes

This block does not have any modes.

### See also

STIn block diagram (Page 2733)
STIn I/Os (Page 2732)
STIn messaging (Page 2732)
STIn error handling (Page 2731)
STIn functions (Page 2731)
Description of STIn (Page 2730)

## 18.8.3 STIn functions

### Functions of STIn

There are no other functions for this block.

### See also

STIn block diagram (Page 2733)
STIn I/Os (Page 2732)
STIn messaging (Page 2732)
STIn error handling (Page 2731)
STIn modes (Page 2731)
Description of STIn (Page 2730)

## 18.8.4 STIn error handling

### STIn error handling

The block does not report any errors.

### See also

STIn block diagram (Page 2733)
STIn I/Os (Page 2732)
STIn messaging (Page 2732)
STIn functions (Page 2731)
STIn modes (Page 2731)
Description of STIn (Page 2730)

## 18.8.5 STIn messaging

### Messaging

This block does not offer messaging.

### See also

STIn block diagram (Page 2733)
STIn I/Os (Page 2732)
STIn error handling (Page 2731)
STIn functions (Page 2731)
STIn modes (Page 2731)
Description of STIn (Page 2730)

## 18.8.6 STIn I/Os

I/Os of STIn

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In | Signal status | BYTE | 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ST_00 | 1 = Bad, device related | BOOL | 0 |
| ST_28 | 1 = Bad, process related | BOOL | 0 |
| ST_60 | 1 = Manipulated value (for example, substitute value, simulation, last valid value) | BOOL | 0 |
| ST_68 | 1 = Unknown, device related | BOOL | 0 |
| ST_78 | 1 = Unknown, process related | BOOL | 0 |
| ST_80 | 1 = Good | BOOL | 0 |
| ST_A4 | 1 = Maintenance request | BOOL | 0 |

### See also

STIn error handling (Page 2731)
STIn block diagram (Page 2733)
Description of STIn (Page 2730)
STIn modes (Page 2731)
STIn functions (Page 2731)
STIn messaging (Page 2732)

## 18.8.7 STIn block diagram

### STIn block diagram

A block diagram is not provided for this block.

### See also

Description of STIn (Page 2730)
STIn modes (Page 2731)
STIn functions (Page 2731)
STIn error handling (Page 2731)
STIn messaging (Page 2732)
STIn I/Os (Page 2732)

# 18.9 STOu - Merging individual binary signals into a signal status

## 18.9.1 Description of STOu

### Object name (type + number) and family

Type + number: FC 374

Family: Convert

### Area of application for STOu

The block is used for the following applications:

- Merging individual binary signals into a signal status

### How it works

The block merges individual binary signals into a `Out` signal status.

If several binary signals are set, the one with the highest priority becomes effective, as described in the section Forming and outputting the signal status for technologic blocks (Page 117) for technologic blocks.

If no binary signal is set, the "Bad, process related" signal status is set.

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

STOu block diagram (Page 2737)

STOu I/Os (Page 2736)

STOu messaging (Page 2736)

STOu error handling (Page 2735)

STOu functions (Page 2735)

STOu modes (Page 2735)

## 18.9.2 STOu modes

### STOu modes

This block does not have any modes.

### See also

STOu block diagram (Page 2737)
STOu I/Os (Page 2736)
STOu messaging (Page 2736)
STOu error handling (Page 2735)
STOu functions (Page 2735)
Description of STOu (Page 2734)

## 18.9.3 STOu functions

### Functions of STOu

There are no other functions for this block.

### See also

STOu block diagram (Page 2737)
STOu I/Os (Page 2736)
STOu messaging (Page 2736)
STOu error handling (Page 2735)
STOu modes (Page 2735)
Description of STOu (Page 2734)

## 18.9.4 STOu error handling

### STOu error handling

The block does not report any errors.

### See also

STOu block diagram (Page 2737)
STOu I/Os (Page 2736)
STOu messaging (Page 2736)
STOu functions (Page 2735)
STOu modes (Page 2735)
Description of STOu (Page 2734)

## 18.9.5 STOu messaging

### Messaging

This block does not offer messaging.

### See also

STOu block diagram (Page 2737)
STOu I/Os (Page 2736)
STOu error handling (Page 2735)
STOu functions (Page 2735)
STOu modes (Page 2735)
Description of STOu (Page 2734)

## 18.9.6 STOu I/Os

**STOu I/Os**

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| ST_00 | 1 = Bad, device related | BOOL | 0 |
| ST_28 | 1 = Bad, process related | BOOL | 0 |
| ST_60 | 1 = Manipulated value (for example, substitute value, simulation, last valid value) | BOOL | 0 |
| ST_68 | 1 = Unknown, device related | BOOL | 0 |
| ST_78 | 1 = Unknown, process related | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| ST_80 | 1 = Good | BOOL | 0 |
| ST_A4 | 1 = Maintenance request | BOOL | 0 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Signal status | BYTE | 16#80 |

### See also

STOu error handling (Page 2735)
STOu block diagram (Page 2737)
STOu messaging (Page 2736)
STOu functions (Page 2735)
STOu modes (Page 2735)
Description of STOu (Page 2734)

## 18.9.7 STOu block diagram

### STOu block diagram

A block diagram is not provided for this block.

### See also

STOu I/Os (Page 2736)
STOu messaging (Page 2736)
STOu error handling (Page 2735)
STOu functions (Page 2735)
STOu modes (Page 2735)
Description of STOu (Page 2734)

# 18.10 MSTIn - Separating the maintenance status into individual status displays

## 18.10.1 Description of MSTIn

### Object name (type + number) and family

Type + number: FB 1858

Family: Convert

### Area of application for MSTIn

The block is used for the following applications:

- Separation of the maintenance status into individual status displays

### How it works

The block separates a maintenance status interconnected to the `In` input parameter into individual status displays.

If the input parameter receives information that at least one value is simulated, for example (`In` = 16#00000003), this is indicated at output parameter `MST_03` with the value 1.

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

MSTIn block diagram (Page 2741)

MSTIn I/Os (Page 2740)

MSTIn messaging (Page 2740)

MSTIn error handling (Page 2739)

MSTIn functions (Page 2739)

MSTIn modes (Page 2739)

## 18.10.2 MSTIn modes

### MSTIn modes

This block does not have any modes.

### See also

MSTIn I/Os (Page 2740)
MSTIn messaging (Page 2740)
MSTIn error handling (Page 2739)
MSTIn functions (Page 2739)
MSTIn block diagram (Page 2741)
Description of MSTIn (Page 2738)

## 18.10.3 MSTIn functions

### Functions of MSTIn

There are no other functions for this block.

### See also

MSTIn block diagram (Page 2741)
MSTIn I/Os (Page 2740)
MSTIn messaging (Page 2740)
MSTIn error handling (Page 2739)
MSTIn modes (Page 2739)
Description of MSTIn (Page 2738)

## 18.10.4 MSTIn error handling

### MSTIn error handling

The block does not report any errors.

### See also

MSTIn block diagram (Page 2741)
MSTIn I/Os (Page 2740)
MSTIn messaging (Page 2740)
MSTIn functions (Page 2739)
MSTIn modes (Page 2739)
Description of MSTIn (Page 2738)

## 18.10.5 MSTIn messaging

### Messaging

This block does not offer messaging.

### See also

MSTIn block diagram (Page 2741)
MSTIn I/Os (Page 2740)
MSTIn error handling (Page 2739)
MSTIn functions (Page 2739)
MSTIn modes (Page 2739)
Description of MSTIn (Page 2738)

## 18.10.6 MSTIn I/Os

### MSTIn I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In | Maintenance status | DWORD | 16#00000000 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| MST_00 | 1 = Good | BOOL | 0 |
| MST_01 | 1 = Passivated | BOOL | 0 |
| MST_02 | 1 = "Out of service" | BOOL | 0 |
| MST_03 | 1 = At least one process value is simulated | BOOL | 0 |
| MST_04 | 1 = "Local mode" | BOOL | 0 |
| MST_05 | 1 = Maintenance requirement | BOOL | 0 |
| MST_06 | 1 = Maintenance request | BOOL | 0 |
| MST_07 | 1 = Maintenance alarm | BOOL | 0 |
| MST_08 | 1 = Unknown | BOOL | 0 |
| MST_09 | 1 = Configuration changed | BOOL | 0 |

### See also

MSTIn block diagram (Page 2741)
MSTIn messaging (Page 2740)
MSTIn error handling (Page 2739)
MSTIn functions (Page 2739)
MSTIn modes (Page 2739)
Description of MSTIn (Page 2738)

## 18.10.7 MSTIn block diagram

### MSTIn block diagram

A block diagram is not provided for this block.

### See also

MSTIn I/Os (Page 2740)
MSTIn messaging (Page 2740)
MSTIn error handling (Page 2739)
MSTIn functions (Page 2739)
MSTIn modes (Page 2739)
Description of MSTIn (Page 2738)

# 18.11 MSTOu - Merging individual status displays into a maintenance status

## 18.11.1 Description of MSTOu

### Object name (type + number) and family

Type + number: FB 1859

Family: Convert

### Area of application for MSTOu

The block is used for the following applications:

- Merging individual status displays into a maintenance status

### How it works

The block merges individual status displays into a maintenance status. If several status displays are set, the status display with the highest number becomes effective.

If status display `MST_03` = 1 is set, for example, this is indicated at output parameter `Out` with the value 16#00000003.

If no status display is set, the `Out` = 16#00 maintenance status is set.

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### See also

MSTOu block diagram (Page 2745)

MSTOu I/Os (Page 2744)

MSTOu messaging (Page 2744)

MSTOu error handling (Page 2744)

MSTOu functions (Page 2743)
MSTOu modes (Page 2743)

## 18.11.2 MSTOu modes

### MSTOu modes

This block does not have any modes.

### See also

MSTOu block diagram (Page 2745)
MSTOu I/Os (Page 2744)
MSTOu messaging (Page 2744)
MSTOu error handling (Page 2744)
MSTOu functions (Page 2743)
Description of MSTOu (Page 2742)

## 18.11.3 MSTOu functions

### Functions of MSTOu

There are no other functions for this block.

### See also

MSTOu block diagram (Page 2745)
MSTOu I/Os (Page 2744)
MSTOu messaging (Page 2744)
MSTOu error handling (Page 2744)
MSTOu modes (Page 2743)
Description of MSTOu (Page 2742)

## 18.11.4 MSTOu error handling

### MSTOu error handling

The block does not report any errors.

### See also

MSTOu block diagram (Page 2745)
MSTOu I/Os (Page 2744)
MSTOu messaging (Page 2744)
MSTOu functions (Page 2743)
MSTOu modes (Page 2743)
Description of MSTOu (Page 2742)

## 18.11.5 MSTOu messaging

### Messaging

This block does not offer messaging.

### See also

MSTOu block diagram (Page 2745)
MSTOu I/Os (Page 2744)
MSTOu error handling (Page 2744)
MSTOu functions (Page 2743)
MSTOu modes (Page 2743)
Description of MSTOu (Page 2742)

## 18.11.6 MSTOu I/Os

### MSTOu I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| MST_00 | 1 = Good | BOOL | 0 |
| MST_01 | 1 = Passivated | BOOL | 0 |
| MST_02 | 1 = "Out of service" | BOOL | 0 |
| MST_03 | 1 = At least one process value is simulated | BOOL | 0 |
| MST_04 | 1 = "Local mode" | BOOL | 0 |
| MST_05 | 1 = Maintenance requirement | BOOL | 0 |
| MST_06 | 1 = Maintenance request | BOOL | 0 |
| MST_07 | 1 = Maintenance alarm | BOOL | 0 |
| MST_08 | 1 = Unknown | BOOL | 0 |
| MST_09 | 1 = Configuration changed | BOOL | 0 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Maintenance status | DWORD | 16#00000000 |

### See also

MSTOu block diagram (Page 2745)
MSTOu messaging (Page 2744)
MSTOu error handling (Page 2744)
MSTOu functions (Page 2743)
MSTOu modes (Page 2743)
Description of MSTOu (Page 2742)

## 18.11.7 MSTOu block diagram

### MSTOu block diagram

A block diagram is not provided for this block.

### See also

MSTOu I/Os (Page 2744)
MSTOu messaging (Page 2744)
MSTOu error handling (Page 2744)

MSTOu functions (Page 2743)
MSTOu modes (Page 2743)
Description of MSTOu (Page 2742)

# 18.12 RealToDw - Converting REAL to DWORD

## 18.12.1 Description of RealToDw

### Object name (type + number) and family

Type + number: FC 390

Family: Convert

### Area of application for RealToDw

The block is used for the following applications:

- Converting a REAL number to a double word (DWORD)

**Note**

Converts the value of a REAL number into a double word (DWORD). You must use the R_DW block (CFC fundamental blocks) to convert the bit character string.

### How it works

With a REAL number between 0 and 4.294967e09 at input `In`, the value is applied and output as a DWORD at the output `Out`.

**Note**

The floating-point numbers in STEP 7 are accurate to 6 decimal places. You can therefore specify a maximum of 6 decimal places for floating-point constants.

If `In` is outside these limits, the error number 30 is output at the `ErrorNum` output and the `Out` is set to these limits.

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for Status parameter

This block does not have the `Status` parameter.

## 18.12.2 RealToDw modes

### Operating modes of RealToDw

This block does not have any modes.

## 18.12.3 ReaLToDw functions

### Functions of RealToDw

There are no other functions for this block.

## 18.12.4 RealToDw error handling

### Error handling of RealToDw

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; block is not processed |
| 0 | There is no error. |
| 30 | `In` cannot be displayed in DWORD format |

## 18.12.5 RealToDw messaging

### Messaging

This block does not offer messaging.

## 18.12.6 RealToDw I/Os

### I/Os of RealToDw

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| In | Analog input value | REAL | 0 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ErrorNum | Error number | INT | -1 |
| Out | Converted output value | DWORD | 0 |
| ST | Status | BYTE | 16#80 |

## 18.12.7 RealToDw block diagram

### Block diagram of RealToDw

A block diagram is not provided for this block.

# 18.13 StateMap - Conversion of other signal states into APL signal states

## 18.13.1 Description of StateMap

**Object name (type + number)**

Type + number: FC 383

Family: Convert

### Area of application of StateMap

This block can map other signal states in an APL signal states.

### How it works

This block converts the following signal states into APL signal states.

| State | ST_In | ST_Out |
|---|---|---|
| Valid value | 16#80 | 16#80 |
| Manipulated value (for example, substitute value, simulation, last valid value) | 16#60 | 16#60 |
| **Substitute value** | **16#48** | **16#60** |
| **Last valid value** | **16#44** | **16#60** |
| Uncertain, device related | 16#68 | 16#68 |
| Uncertain, process related | 16#78 | 16#78 |
| **Unknown, device related, range violation** | **16#54** | **16#68** |
| Maintenance request | 16#A4 | 16#A4 |
| Invalid value (-STOP) | 16#00 | 16#00 |
| Uncertain, device related | Otherwise | **16#68** |

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

## 18.13.2 StateMap modes

**StateMap modes**

This block does not have any operating modes.

## 18.13.3 StateMap functions

**Functions of StateMap**

There are no other functions for this block.

## 18.13.4 StateMap error handling

**StateMap error handling**

This block does not have any error handling.

## 18.13.5 StateMap messaging

**Messaging**

This block does not offer messaging.

## 18.13.6 StateMap I/Os

**I/Os of StateMap**

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Meaning | Data type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| ST_In | Signal status input | BYTE | 16#80 |

### Output parameters

| Parameter | Meaning | Data type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ST_Out | Signal status output | BYTE | 16#80 |

## 18.13.7 StateMap block diagram

### StateMap block diagram

A block diagram is not provided for this block.

# 18.14 StruCmIn - Splitting a structure to outputs

## 18.14.1 Description of StruCmIn

### Object name (type + number) and family

Type + number: FC 446

Family: Convert

### Area of application for StruCmIn

The block is used for the following applications:

- Splitting the structure of process values to separate outputs

The block can be used together with StruCmOu to interconnect a group of process values via only one connection.

### How it works

The block separates a structure interconnected to the `In` input parameter to the separate outputs `AnaVal[x]`, `DigVa[x]`, `CntVal1`, `Scale[n]`, `Int[n]` and `DwVal1`, x=1..6, n=1..2.

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for parameter `Status`

This block does not have the `Status` parameter

## 18.14.2 StruCmIn modes

### StruCmIn Modes

This block does not have any modes.

## 18.14.3 StruCmIn functions

### Functions of StruCmIn

There are no other functions for this block.

## 18.14.4 StruCmIn error handling

### StruCmIn Error handling

The block does not report any errors.

## 18.14.5 StruCmIn messaging

### StruCmIn Messaging

This block does not offer messaging.

## 18.14.6 StruCmIn I/Os

I/Os of StruCmIn

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `In.Anaval1 ... In.Anaval6` | Analog value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>0.0<br>16#00 |
| `In.CntVal1` | Counter value | STRUCT<br>• Value: DINT<br>• ST: BYTE | -<br>0<br>16#80 |
| `In.Digval1 ... In.Digval6` | Digital value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>0<br>16#00 |
| `In.IntVal1 ... In.IntVal2` | Integer value | INT | 0 |
| `In.Scale1H` | Scale high | REAL | 100.0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| In.Scale1L | Scale low | REAL | 0.0 |
| In.Scale2H | Scale high | REAL | 100.0 |
| In.Scale2L | Scale low | REAL | 0.0 |
| In.DwVal1 | Dword value | DWORD | 16#00000000 |
| EN | 1 = Called block will be processed | BOOL | 1 |
| In | - | STRUCT | - |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Anaval1 ... Anaval6 | Analog value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>0.0<br>16#80 |
| Digval1 ... Digval6 | Digital value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>0.0<br>16#80 |
| CntVal1 | Counter value | STRUCT<br>• Value: DINT<br>• ST: BYTE | -<br>0<br>16#80 |
| ENO | - | BOOL | 0 |
| Scale1.High | Scale high | REAL | 100.0 |
| Scale1.Low | Scale low | REAL | 0.0 |
| Scale2.High | Scale high | REAL | 100.0 |
| Scale2.Low | Scale low | REAL | 0.0 |
| IntVal1 ... IntVal2 | Integer value | INT | 0 |
| DwVal1 | Dword value | DWORD | 16#00000000 |

## 18.14.7 StruCmIn block diagram

### StruCmIn block diagram

A block diagram is not provided for this block.

# 18.15 StruCmOu - Merging inputs to a structure

## 18.15.1 Description of StruCmOu

### Object name (type + number) and family

Type + number: FC 447

Family: Convert

### Area of application for StruCmOu

The block is used for the following applications:

- Merging a group of process values to a structure

The block can be used together with StruCmIn to interconnect a group of process values via only one connection.

### How it works

The block merges the inputs `AnaVal[x]`, `DigVa[x]`, `CntVal1`, `Scale[n]`, `Int[n]` and `DwVal1` to a structure (`Out`), x=1..6, n=1..2

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for parameter `Status`

This block does not have the `Status` parameter.

## 18.15.2 StruCmOu modes

### StruCmOu modes

This block does not have any modes.

## 18.15.3 StruCmOu functions

### StruCmOu functions

There are no other functions for this block.

## 18.15.4 StruCmOu error handling

### StruCmOu error handling

This block does not report any errors.

## 18.15.5 StruCmOu messaging

### StruCmOu messaging

This block does not offer messaging.

## 18.15.6 StruCmOu I/Os

**I/Os of StruCmOu**

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `AnaVal1 ... AnaVal6` | Analog value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>0.0<br>16#80 |
| `DigVal1 ... DigVal6` | Digital value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>0<br>16#80 |
| `DwVal1` | Dword value | DWORD | 16#00000000 |
| `CntVal1` | Counter value | STRUCT<br>• Value: DINT<br>• ST: BYTE | -<br>0<br>16#80 |
| `EN` | 1 = Called block will be processed | BOOL | 1 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| `IntVal1 ... IntVal2` | Integer value | INT | 0 |
| `Scale1.Low` | Scale low | REAL | 0.0 |
| `Scale2.Low` | Scale low | REAL | 0.0 |
| `Scale1.High` | Scale high | REAL | 100.0 |
| `Scale2.High` | Scale high | REAL | 100.0 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `Out.AnaVal` | Analog value | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>0.0<br>16#80 |
| `Out.DigVal1 ...Out.DigVal6` | Digital value | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>0<br>16#00 |
| `Out.CntVal1` | Count value | STRUCT<br>• Value: DINT<br>• ST: BYTE | -<br>0<br>16#00 |
| `Out.IntVal1 ... Out.IntVal2` | Integer value | INT | 0 |
| `Out.Scale1H` | Scale high | REAL | 0.0 |
| `Out.Scale1L` | Scale low | REAL | 0.0 |
| `Out.Scale2H` | Scale high | REAL | 0.0 |
| `Out.Scale2L` | Scale low | REAL | 0.0 |
| `Out.DwVal1` | Dword value | DWORD | 16#00000000 |
| `Out` | - | STRUCT | - |
| `ENO` | - | BOOL | 0 |

## 18.15.7 StruCmOu block diagram

### StruCmOu block diagram

A block diagram is not provided for this block.

# 18.16 StruCnIn - Separating a counter structured variable

## 18.16.1 Description of StruCnIn

### Object name (type + number) and family

Type + number: FC 458

Family: Convert

### Area of application for StruCnIn

The block is used for the following applications:

- Separation of an analog value with a structure into a variable of the DINT data type and a signal status

### How it works

The block separates a counter value with a structure interconnected to the `In` input parameter into a variable (`Out`) of the DINT data type and a (ST) signal status.

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for parameter `Status`

This block does not have the `Status` parameter.

## 18.16.2 StruCnIn modes

### StruCnIn modes

This block does not have any modes.

## 18.16.3 StruCnIn functions

### StruCnIn functions

There are no other functions for this block.

## 18.16.4 StruCnIn error handling

### StruCnIn error handling

The block does not report any errors.

## 18.16.5 StruCnIn messaging

### Messaging

This block does not offer messaging.

## 18.16.6 StruCnIn I/Os

### StruCnIn I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In | Counter input value with structure | STRUCT<br>• Value: DINT<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| ST | Signal status | BYTE | 16#80 |
| Value | Counter value without signal state | REAL | 0.0 |

## 18.16.7 StruCnIn block diagram

### StruCnIn block diagram

A block diagram is not provided for this block.

# 18.17 StruCnOu - Creating a structured counter variable

## 18.17.1 Description of StruCnOu

### Object name (type + number) and family

Type + number: FC 459

Family: Convert

### Area of application for StruCnOu

The block is used for the following applications:

- Merging a variable of the DINT data type and a signal status into an analog process value.

### How it works

The block merges a counter value with a structure interconnected to the `In` input parameter into a variable (`Out`) of the DINT data type and a (ST) signal status.

### Configuration

Use the CFC editor to install the block in any OB.

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for parameter `Status`

This block does not have the `Status` parameter.

## 18.17.2 StruCnOu modes

### StruCnOu

This block does not have any modes.

## 18.17.3 StruCnOu functions

### StruCnOu functions

There are no other functions for this block.

## 18.17.4 StruCnOu error handling

### StruCnIn error handling

The block does not report any errors.

## 18.17.5 StruCnOu messaging

### Messaging

This block does not offer messaging.

## 18.17.6 StruCnOu I/Os

### StruCnOu I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| Value | Counter value without signal state | DINT | 0 |
| ST | Signal state | BYTE | 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| Bad | 1 = (16#00 <= ST <= 16#3F) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| Out | Counter output value | STRUCT<br>• Value: DINT<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

## 18.17.7 StruCnOu block diagram

### StruCnIn block diagram

A block diagram is not provided for this block.

# Maintenance blocks 19

## 19.1 MuxMST - Determination of the worst maintenance status

### 19.1.1 Description of MuxMST

#### Object name (type + number) and family

Type + number: FB 1861

Family: Maint

#### Area of application for MuxMST

The block is used for the following applications:

- Determination of the worst maintenance status (from a maximum of 10 statuses)

#### How it works

The block determines the worst of several maintenance states. Each status is made available to the block via interconnection to the `Inx` input parameter (x = 01 ... 10) . The states are compared and the highest value written to the `Out` output parameter.

#### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

#### Startup characteristics

During startup, the `Inx` input parameters (x = 01 ... 10) and output parameter are reset to their defaults.

#### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

#### See also

MuxMST block diagram (Page 2768)

MuxMST I/Os (Page 2767)

MuxMST messaging (Page 2767)

MuxMST error handling (Page 2767)
MuxMST functions (Page 2766)
MuxMST modes (Page 2766)

## 19.1.2 MuxMST modes

### MuxMST operating modes

This block does not have any modes.

### See also

MuxMST block diagram (Page 2768)
MuxMST I/Os (Page 2767)
MuxMST messaging (Page 2767)
MuxMST error handling (Page 2767)
MuxMST functions (Page 2766)
Description of MuxMST (Page 2765)

## 19.1.3 MuxMST functions

### Functions of MuxMST

This block provides no other functions.

### See also

MuxMST block diagram (Page 2768)
MuxMST I/Os (Page 2767)
MuxMST messaging (Page 2767)
MuxMST error handling (Page 2767)
MuxMST modes (Page 2766)
Description of MuxMST (Page 2765)

## 19.1.4 MuxMST error handling

### MuxMST error handling

The block does not report any errors.

### See also

MuxMST block diagram (Page 2768)
MuxMST I/Os (Page 2767)
MuxMST messaging (Page 2767)
MuxMST functions (Page 2766)
MuxMST modes (Page 2766)
Description of MuxMST (Page 2765)

## 19.1.5 MuxMST messaging

### Messaging

This block does not offer messaging.

### See also

MuxMST block diagram (Page 2768)
MuxMST I/Os (Page 2767)
MuxMST error handling (Page 2767)
MuxMST functions (Page 2766)
MuxMST modes (Page 2766)
Description of MuxMST (Page 2765)

## 19.1.6 MuxMST I/Os

### MuxMST I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In01 | Input for signal of maintenance status 0 | DWORD | 16#00000000 |
| In02 | Input for signal of maintenance status 1 | DWORD | 16#00000000 |
| In03 | Input for signal of maintenance status 2 | DWORD | 16#00000000 |
| In04 | Input for signal of maintenance status 3 | DWORD | 16#00000000 |
| In05 | Input for signal of maintenance status 4 | DWORD | 16#00000000 |
| In06 | Input for signal of maintenance status 5 | DWORD | 16#00000000 |
| In07 | Input for signal of maintenance status 6 | DWORD | 16#00000000 |
| In08 | Input for signal of maintenance status 7 | DWORD | 16#00000000 |
| In09 | Input for signal of maintenance status 8 | DWORD | 16#00000000 |
| In10 | Input for signal of maintenance status 9 | DWORD | 16#00000000 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Output signal with the worst signal status | DWORD | 16#00000000 |

### See also

MuxMST block diagram (Page 2768)
MuxMST messaging (Page 2767)
MuxMST error handling (Page 2767)
MuxMST functions (Page 2766)
MuxMST modes (Page 2766)
Description of MuxMST (Page 2765)

## 19.1.7 MuxMST block diagram

### MuxMST block diagram

A block diagram is not provided for this block.

### See also

MuxMST I/Os (Page 2767)
MuxMST messaging (Page 2767)
MuxMST error handling (Page 2767)

MuxMST functions (Page 2766)
MuxMST modes (Page 2766)
Description of MuxMST (Page 2765)

# 19.2 MuxST- Determination of the worst signal status

## 19.2.1 Description of MuxST

### Object name (type + number) and family

Type + number: FB 1862

Family: Maint

### Area of application for MuxST

The block is used for the following applications:

- Determination of the worst signal status (from a maximum of 10 statuses)

### How it works

The block determines the worst of several signal states. Each status is made available to the block via interconnection to the `Inx` input parameter (x = 1 ... 10). The states are compared and the value with the highest priority is written to the `Out` output parameter.

Also refer to the section Forming and outputting the signal status for blocks with configurable status prioritization (Page 122) for information on priorities.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

During startup, the `Inx` input parameters (x = 1 ... 10) and output parameter are reset to their defaults.

### Status word allocation for `Status1` parameter

This block does not have the `Status` parameter.

### See also

MuxST block diagram (Page 2774)

MuxST I/Os (Page 2773)

MuxST messaging (Page 2772)

MuxST error handling (Page 2772)

MuxST functions (Page 2771)
MuxST modes (Page 2771)

## 19.2.2 MuxST modes

### MuxST operating modes

This block does not have any modes.

### See also

MuxST block diagram (Page 2774)
MuxST I/Os (Page 2773)
MuxST messaging (Page 2772)
MuxST error handling (Page 2772)
MuxST functions (Page 2771)
Description of MuxST (Page 2770)

## 19.2.3 MuxST functions

### Functions of MuxST

The functions for this block are listed below.

### Selecting signals for processing

Using the `SelInput` input parameter, you select the number of interconnectable input parameters to be used for processing in the block. The selected number denotes the input parameters used `In1` ... `Inn`.

If, for example, you set this input parameter to `SelInput` = 3, the input parameters `In1`, `In2` and `In3` will be used for processing.

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for blocks with configurable status prioritization (Page 122).

### See also

MuxST block diagram (Page 2774)
MuxST I/Os (Page 2773)

MuxST messaging (Page 2772)
MuxST error handling (Page 2772)
MuxST modes (Page 2771)
Description of MuxST (Page 2770)

## 19.2.4 MuxST error handling

### MuxST error handling

The block does not report any errors.

### See also

MuxST block diagram (Page 2774)
MuxST I/Os (Page 2773)
MuxST messaging (Page 2772)
MuxST functions (Page 2771)
MuxST modes (Page 2771)
Description of MuxST (Page 2770)

## 19.2.5 MuxST messaging

### Messaging

This block does not offer messaging.

### See also

MuxST block diagram (Page 2774)
MuxST I/Os (Page 2773)
MuxST error handling (Page 2772)
MuxST functions (Page 2771)
MuxST modes (Page 2771)
Description of MuxST (Page 2770)

## 19.2.6 MuxST I/Os

### MuxST I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| In1 | Input for signal of signal status 0 | BYTE | 16#80 |
| In2 | Input for signal of signal status 1 | BYTE | 16#80 |
| In3 | Input for signal of signal status 2 | BYTE | 16#80 |
| In4 | Input for signal of signal status 3 | BYTE | 16#80 |
| In5 | Input for signal of signal status 4 | BYTE | 16#80 |
| In6 | Input for signal of signal status 5 | BYTE | 16#80 |
| In7 | Input for signal of signal status 6 | BYTE | 16#80 |
| In8 | Input for signal of signal status 7 | BYTE | 16#80 |
| In9 | Input for signal of signal status 8 | BYTE | 16#80 |
| In10 | Input for signal of signal status 9 | BYTE | 16#80 |
| SelInput* | Selection of the input parameter for forming the worst signal status | INT | 2 |
| SelPrio* | Setting of the prioritization for forming the worst signal status | INT | 0 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| Out | Output of the worst signal status | BYTE | 16#80 |

### See also

MuxST block diagram (Page 2774)

MuxST messaging (Page 2772)

MuxST error handling (Page 2772)

MuxST functions (Page 2771)

MuxST modes (Page 2771)

Description of MuxST (Page 2770)

## 19.2.7 MuxST block diagram

### MuxST block diagram

A block diagram is not provided for this block.

### See also

MuxST I/Os (Page 2773)
MuxST messaging (Page 2772)
MuxST error handling (Page 2772)
MuxST functions (Page 2771)
MuxST modes (Page 2771)
Description of MuxST (Page 2770)

# 19.3 STRep - Status display of block groups

## 19.3.1 Description of STRep

### Object name (type + number) and family

Type + number: FB 1801

Family: MAINT

### Area of application of STRep

The STRep block is used for displaying the status of a group of blocks that is designed for hiding messages automatically.

### How it works

The block has 32 `STRUCT` type inputs that describe defined states. Depending on which `StateX` input is set, it will be output at the INT output `QSTATE`. If several `StateX` inputs are set, the most significant is output and the `MPSA` output is set.

The status of the `StateX` inputs does not influence the result at `QSTATE`.

**Note**

The output `QSTATE` has the attribute S7_enum := 'Operating State'. In your project under the folder 'Shared Declarations' you can use the enumeration 'Operating State' to define the text for the enumeration value. If you want to use your own enumeration, you have to adapt the S7_enum attribute at the output `QSTATE`.

### Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100).

### Startup characteristics

The block does not have any startup characteristics.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

## 19.3.2 STRep modes

### Operating modes of STRep

This block does not have any modes.

## 19.3.3 STRep functions

### Functions of STRep

There are no other functions for this block.

## 19.3.4 STRep error handling

### Error handling of STRep

The block does not report any errors.

## 19.3.5 STRep messaging

### Messaging

This block does not offer messaging.

## 19.3.6 STRep I/Os

I/Os of STRep

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `State1 ... State32` | Process status 1 ... 32 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `QERR` | 1 = Error | BOOL | 1 |
| `QSTATE` | Process status | INT | 0 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| MPSA | 1 = More than one process status is active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 1<br>• 16#80 |
| QSTATE | Process status 0...32<br>With attribute:<br>S7_enum := 'Operating State' | INT | 0 |

## 19.3.7 STRep block diagram

### Block diagram of STRep

A block diagram is not provided for this block.

# 19.4 AssetM - Process variable monitoring for violation of limits

## 19.4.1 Description of AssetM

### Object name (type + number) and family

Type and number: FB 1840

Family: Maint

### Area of application of AssetM

The block is used for the following applications:

Monitoring of up to 3 analog process variables for exceeding 3 limits respectively. It reports the maintenance status of the process variables:

- When overshooting a limit
- Via the device-based signal status or
- Via binary message inputs

### How it works

The block monitors up to 3 inputs for high and low violations of three limits respectively. Upon reaching or exceeding a limit, the respective output is set and a corresponding message is generated.

The three process variables are equivalent for limit monitoring as long as the overshoot of a limit (e.g. maintenance request) is reported by one process variable. A new message will only be created again if all process variables have undershot this limit at least for one cycle.

The monitoring of individual limits can be deactivated.

In addition to limit monitoring, the signal status of the individual process variables is also analyzed. The device-specific signal status of a process value triggers the corresponding message that is also used by the limit monitoring.

### Configuration

In the CFC editor, install the block in a cyclic interrupt OB (e.g. OB32). The block is also installed automatically in the startup OB (OB 100).

### Startup characteristics

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

### Binary message inputs

| | | |
|---|---|---|
| Message1 | Bad, maintenance alarm | Message (S) requires acknowledgement |
| Message2 | Uncertain, maintenance request | Message (F) requires acknowledgement |
| Message3 | Good, maintenance required | Message (M) requires acknowledgement |
| Message4 | Bad, local operation/functional check | Message (S) requires acknowledgement |
| Message5 | Uncertain, simulation | Message (SA) does not require acknowledgement |
| Message6 | Bad, device out of service | Message (SA) does not require acknowledgement |
| Message7 | bad, passivated | Message (SA) does not require acknowledgement |

For each PVx, there is an input (PVx_Rst) and an output (P_PVx_Rst) that can be used to reset a technologic block.

The statuses are created with ALARM_8P for messages requiring acknowledgment and with NOTIFY_8P for those not requiring acknowledgment. The message function can be disabled by setting MsgLock = 0. In this case, MS = 8 is set.

The detailed diagnostics is shown in the diagnostic view of the faceplate via the inputs Diag1 to Diag16.

If one of the inputs is set to 1, then the status display is shown in front of the corresponding text.

The texts for the relevant inputs Diag1 to Diag16 are entered in the parameter data of the EDD for the relevant instance (see also section PLT ID).

If one of the inputs Diag1 to Diag16 is set to 1, the explanatory text "Additional status available" is output if an internal interrupt is triggered through PV0, PV1 or PV2.

### PLT-ID

Die PLT-ID is a connection parameter between a PDM object (parameter data EDD) and the faceplate in the maintenance station. The PLT-ID is linked to the PDM object.

The PDM object is generated in the SIMATIC Manager as follows:

1. Select **View > Process Device Plant View** in SIMATIC Manager.
2. In SIMATIC PDM, select a project in which you want to insert the AssetM block.
3. Select **File > New object**.
4. Right-click the inserted tag object and select **Device selection (Reassign)...** from the context menu.
5. In the tree structure **Devices > DATA_OBJECTS > CFC >**, select AssetMon and click **OK**.
6. Right-click the tag object and select **Open object** from the context menu, and then enter all necessary data in the parameter assignment screen form.
7. Select **File > Save** and close the parameter assignment screen form.

You can then assign parameters for the generated PLT-ID at the associated parameter "PLT_ID".

**Note**

The PLT-IDs cannot be changed or deleted individually.

### Creating the maintenance status (MS)

MS depends on:

- From the signal status of the signals `PV0`, `PV1` and `PV2`
- from the binary message inputs (external `MS`)
- from the interconnectable input `MS_In` (external maintenance state)
- The interconnectable input `STATUS`. The process-related errors have no effect; only the device-based errors have an effect.

Of all these events the highest priority event will be displayed in the MS.

The 16 `Diagx` binary inputs do not have any influence on the MS; they are used only for visualization of the detail diagnostics in the diagnostics view of the faceplate.

**Note**

The table is valid exclusively for the MS and not for the signal status displays of the individual process values; these are created exclusively by the `ST` parameters.

The priority is similar to the `MS` coding. Therefore, the following applies: The greater the value of the `MS`, the higher the priority.

| PVx.Value | PVx.ST | MS_In | Event | STATUS | Messa-gex.Value | O_MS |
|---|---|---|---|---|---|---|
| - | - | 9 | Configuration change | 16#84 .. 87 | - | 9 |
| - | - | 8 | Untested/unknown | - | - | 8 |
| PVx.Value >= PVx_AH | 16#00 | 7 | bad, maintenance alarm | 16#00 .. 1B, 16#24 .. 27, 16#44 .. 4B | x = 1 | 7 |
| PVx_AH > PVx.Value >= PVx_DH | 16#68 | 6 | uncertain, maintenance request | 16#40 .. 43, 16#50 .. 5F, 16#64 .. 6B, 16#A8 .. AB | x = 2 | 6 |
| PVx_DH > PVx.Value >= PVx_RH | 16#A4 | 5 | good, maintenance required | 16#A4 .. A7 | x = 3 | 5 |
| - | - | 4 | bad, local operation/functional check | 16#3C .. 3F | x = 4 | 4 |
| - | 16#60 | 3 | Uncertain, manipulated value (for example, substitute value, simulation, last valid value) | 16#60 .. 63, 16#70 .. 73 | x = 5 | 3 |
| - | - | 2 | bad, device out of service | 16#1C .. 1F | x = 6 | 2 |

| PVx.Value | PVx.ST | MS_In | Event | STATUS | Messa-gex.Value | O_MS |
|---|---|---|---|---|---|---|
| - | - | 1 | bad, passivated | 16#23 | x = 7 | 1 |
| PVx_RH > PVx.Value | 16#28, 16#78 | 0 | process-related fault | 16#28 .. 2B, 16#78 .. 7B, 16#A0 .. A3 | - | |

## 19.4.2 AssetM modes

### Operating modes of AssetM

This block does not have any modes.

## 19.4.3 AssetM functions

### Functions of AssetM

The functions for this block are listed below.

### Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Configurable functions using the Feature I/O (Page 139) section. The following functionality is available for this block at the relevant bits:

| Bit | Function |
|---|---|
| 22 | Update acknowledgment and error status of the message call (Page 171) |
| 28 | Disabling operating points (Page 156) |

### Forming the signal status for blocks

This block provides the standard function Forming and outputting the signal status for interlock blocks (Page 124).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `PV0.ST`
- `PV1.ST`
- `PV2.ST`

## 19.4.4 AssetM error handling

### Error handling of AssetM

Refer to the section Error handling (Page 128) in the basic instructions for the error handling of all blocks.

The following errors can be displayed for this block:

- Error numbers

### Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

| Error number | Meaning of the error number |
|---|---|
| -1 | Predefined value when inserting the block; block is not processed |
| 0 | There is no error. |

## 19.4.5 AssetM messages

### Messaging

The following messages can be generated for this block:

- Process control fault
- Process messages

### Process control fault

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| `MsgEvId1` | SIG 1 | AS process control message - fault | Maintenance alarm<br>@4W%t#AssetM_TXT@ |
| | SIG 2 | AS process control message - error | Maintenance demanded<br>@4W%t#AssetM_TXT@ |
| | SIG 3 | Preventive maintenance - General | Maintenance required<br>@4W%t#AssetM_TXT@ |
| | SIG 4 | Reserved | |
| | SIG 5 | Reserved | |
| | SIG 6 | Reserved | |
| | SIG 7 | AS process control message - fault | Device out of service |
| | SIG 8 | Reserved | |

### Process messages

| Message instance | Message identifier | Message class | Event |
|---|---|---|---|
| MsgEvId2 | 1 | Status AS | Passivated |
| | 2 | Reserved | |
| | 3 | Status AS | Manipulated value (for example, substitute value, simulation, last valid value) |
| | 4 | Status AS | Manipulated value (for example, substitute value, simulation, last valid value) |
| | 5 | Status AS | Configuration change |
| | 6 | Status AS | Process-related fault |
| | 7 | Reserved | |
| | 8 | Reserved | |

### Associated values

| Message block ALARM 8P | Associated value | Meaning |
|---|---|---|
| MsgEvId1 | 4 | Text number from AssetM_TXT |

### System text library AssetM

| Index | Text |
|---|---|
| 1 | Further status available |
| 2 | No further status available |

## 19.4.6 AssetM I/Os

### I/Os of AssetM

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

## Input parameters

| I/O (parameter) | Meaning | Type | Default |
|---|---|---|---|
| Diagx | Asset detail diagnosis (x = 1 to 16) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Feature | I/O for additional functions (Page 2781) | STRUCT<br>• Bit 0: BOOL<br>• ...<br>• Bit 31: BOOL | -<br>• 0<br>• 0<br>• 0 |
| Message1 | 1 = message: bad, maintenance alarm | BOOL | 0 |
| Message2 | 1 = message: uncertain, maintenance request | BOOL | 0 |
| Message3 | 1 = message: good, maintenance required | BOOL | 0 |
| Message4 | 1 = message: bad, local operation/functional check | BOOL | 0 |
| Message5 | 1 = message: uncertain, simulation | BOOL | 0 |
| Message6 | 1 = message: bad, out of service | BOOL | 0 |
| Message7 | 1 = message: passivated | BOOL | 0 |
| MS* | Maintenance status | DWORD | 16#000000 00 |
| MsgEvId1 | Message event ID 1 | DWORD | 16#000000 24 |
| MsgEvId2 | Message event ID 2 | DWORD | 16#000000 25 |
| MsgLock | 1 = Suppress process messages.<br>Refer also to the Suppressing messages using the MsgLock parameter (Page 231) chapter for more on this. | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| MS_Release | Release for maintenance | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| Ms_In | External interconnectable MS | DWORD | 16#000000 00 |
| PLT_ID | Asset ID of EDD | DWORD | 16#000000 00 |
| PV_Hyst | PV_Hysteresis for messages | REAL | 5.0 |
| PVx_AH | Limit PVx (x=0, 1, 2) maintenance alarm | REAL | 100.0 |
| PVx_AH_EN | 1 = suppress monitoring of the limit PVx (x = 0, 1, 2), maintenance alarm exceeded | BOOL | 0 |
| PVx_DH | Limit PVx (x = 0, 1, 2) maintenance request | REAL | 100.0 |
| PVx_DH_EN | 1 = suppress monitoring of the limit PVx (x = 0, 1, 2), maintenance request exceeded | BOOL | 0 |
| PVx_RH | Limit PVx (x = 0, 1, 2) maintenance required | REAL | 100.0 |
| PVx_RH_EN | 1 = suppress monitoring of the limit PVx (x = 0, 1, 2), maintenance required exceeded | BOOL | 0 |

| I/O (parameter) | Meaning | Type | Default |
|---|---|---|---|
| PVx | Process value PVx (x = 0, 1, 2) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PVx_Rst* | 1 = Reset PVx (x = 0, 1, 2) | BOOL | 0 |
| PVxUnit | Unit of the process value | INT | 0 |
| RunUpCyc | Number of run-up cycles | INT | 3 |
| Status | External status | BYTE | 16#80 |

* Values can be written back to these inputs during processing of the block by the block algorithm.

### Output parameters

| I/O (parameter) | Meaning | Type | Default |
|---|---|---|---|
| ErrorNum | Program error | INT | -1 |
| MsgAckn1 | Message recognized | WORD | 16#0000 |
| MsgErrx | Message errors occurred (x =1, 2) | WORD | 16#0000 |
| MsgStatx | Message error information (x = 1, 2) | WORD | 16#0000 |
| O_MS | Maintenance status | DWORD | 16#00000000 |
| PVx_Diff | Differential value to the next expected alarm (x = 0, 1, 2) | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| PVx_AH_AcT | 1 = Limit PVx (x = 0, 1, 2), maintenance alarm exceeded | BOOL | 0 |
| PVx_DH_AcT | 1 = Limit PVx (x = 0, 1, 2), maintenance required exceeded | BOOL | 0 |
| PVx_RH_AcT | 1 = Limit PVx (x = 0, 1, 2), maintenance request exceeded | BOOL | 0 |
| P_PVx_Rst | 1 = Reset PVx (x = 0, 1, 2) | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| ST_Worst | Worst signal status of the inputs PVx (x = 0, 1, 2) | BYTE | 16#80 |

### See also

AssetM functions (Page 2781)

## 19.4.7 AssetM block diagram

### Block diagram of ASSETM

A block diagram is not provided for this block.

# System blocks

# 20

## 20.1 AddInt64 - Addition of two 64-bit integer variables

### 20.1.1 Description of AddInt64

#### Object name (type + number) and family

Type + number: FC 353

Family: System

#### Area of application for AddInt64

The block is used for the following applications:

- Addition of two 64-bit integer variables

This block is used by the PIDConR block for calculations that are twice as accurate (64-bit number format).

Therefore this block will not be described in detail.

# 20.2 AddR64 - Addition of two 64-bit REAL variables

## 20.2.1 Description of AddR64

### Object name (type + number) and family

Type + number: FC 354

Family: System

### Area of application for AddR64

The block is used for the following applications:

- Addition of two 64-bit REAL variables

This block is used by the PIDConR block for calculations that are twice as accurate (64-bit number format).

Therefore this block will not be described in detail.

# 20.3 DiToInt64 - Converting from DINT to Int64

## 20.3.1 Description of DiToInt64

### Object name (type + number) and family

Type + number: FC 357

Family: System

### Area of application for DiToInt64

The block is used for the following applications:

- Converting from DINT to Int64

This block is used by the PIDConR block for calculations that are twice as accurate (64-bit number format).

Therefore this block will not be described in detail.

# 20.4 Int64ToDi - Converting from Int64 to DINT

## 20.4.1 Description of Int64ToDi

### Object name (type + number) and family

Type + number: FC 359

Family: System

### Area of application for Int64ToDi

The block is used for the following application:

- Converting from Int64 to DINT

This block will not be described in detail.

# 20.5 MemR256 - Increasing the number of internal previous values

## 20.5.1 Description of MemR256

### Object name (type + number)

Object name (type + number) and family

Type and number: FB 1930

Family: System

### Area of application

The block is used for the following applications:

- Increasing the internal previous values in the MeanTime block.

Therefore this block will not be described in detail.

# 20.6 NegInt64 - Negation of an Int64 variable

## 20.6.1 Description of NegInt64

### Object name (type + number) and family

Type + number: FC 362

Family: System

### Area of application for NegInt64

The block is used for the following applications:

- Negation of an Int64-variable

This block is used by the PIDConR block for calculations that are twice as accurate (64-bit number format).

Therefore this block will not be described in detail.

# 20.7 NegR64 - Negation of a Real64 variable

## 20.7.1 Description of NegR64

### Object name (type + number) and family

Type + number: FC 363

Family: System

### Area of application for NegR64

The block is used for the following application:

- Negation of an Real64-variable

This block will not be described in detail.

# 20.8 PIDCoefR - Calculation of coefficients

## 20.8.1 Description of PIDCoefR

### Object name (type + number) and family

Type + number: FC 366

Family: System

### Area of application for PIDCoefR

The block is used for the following applications:

- Calculation of coefficients of discrete-time difference equations for PIDConR from the continuous-time input parameters (e.g. `TI`, `TD`)

This block is used by the PIDConR block for calculations that are twice as accurate (64-bit number format).

Therefore this block will not be described in detail.

# 20.9 R64ToReal - Converting Real64 to REAL

## 20.9.1 Description of R64ToReal

### Object name (type + number) and family

Type + number: FC 367

Family: System

### Area of application for R64ToReal

The block is used for the following applications:

- Converting from Real64 to REAL (32 Bit)

This block is used by the PIDConR block for calculations that are twice as accurate (64-bit number format).

Therefore this block will not be described in detail.

# 20.10 RealToR64 - Converting REAL to Real64

## 20.10.1 Description of RealToR64

### Object name (type + number) and family

Type + number: FC 368

Family: System

### Area of application for RealToR64

The block is used for the following applications:

- Converting from REAL (32 Bit) to Real64

This block is used by the PIDConR block for calculations that are twice as accurate (64-bit number format).

Therefore this block will not be described in detail.

# 20.11 SelST16 - Output of the best or worst signal status

## 20.11.1 Description of SelST16

### Object name (type + number) and family

Type and number: FC 369

Family: System

### Area of application for SelST16

The block is used for the following applications:

- Output the best or worst signal status

This block is used by technologic blocks which edit and output a signal status.

Therefore this block will not be described in detail.

# 20.12 ShLeInt64 - Left shift of an Int64 variable

## 20.12.1 Description of ShLeInt64

### Object name (type + number) and family

Type + number: FC 370

Family: System

### Area of application for ShLeInt64

The block is used for the following applications:

- Left shift of a (positive) Int64- variable

This block is used by the PIDConR block for calculations that are twice as accurate (64-bit number format).

Therefore this block will not be described in detail.

# 20.13 ShRiInt64 - Right shift of an Int64 variable

## 20.13.1 Description of ShRiInt64

### Object name (type + number) and family

Type + number: FC 371

Family: System

### Area of application for ShRiInt64

The block is used for the following applications:

- Right shift of a (positive) Int64- variable

This block is used by the PIDConR block for calculations that are twice as accurate (64-bit number format).

Therefore this block will not be described in detail.

# 20.14 PIDKernR - Calculation of the manipulated variable

## 20.14.1 Description of PIDKernR

### Object name (type + number) and family

Type and number: FB 1877

Family: System

### Area of application for PIDKernR

The block is used for the following applications:

- Calculation of the manipulated variable in PIDConR

This block is used by the PIDConR block for calculations that are twice as accurate (64-bit number format).

Therefore this block will not be described in detail.

# Communication blocks 21

## 21.1 Snd_DigVal - Send Boolean values including quality code (Snd_DigVal)

### 21.1.1 Description of Snd_DigVal

#### Object name (type + number) and family

Type + number: FB 1891

Family: COMM

#### Area of application for Snd_DigVal

The "Snd_DigVal" block represents a simple user interface to SFB 12 "BSEND". The "Snd_DigVal" block sends 128 DigVal structures via an MPI, PROFIBUS or Industrial Ethernet connection to another CPU. This CPU calls the function block "Rcv_DigVal" (FB 1892) of the PCS 7 Advanced Process Library to receive the data.

#### How it works

Data transfer is initiated by calling the block with the value 1 at the `SendEn` control input. While data is being sent, `SendAct` is set to 1.

If the job is completed without errors, `SendDone` is set to 1. `SendErr` = 1 if an error occurs. A new job is then automatically triggered with the current data until all data have been successfully sent.

If input `SendEn` = 0, incomplete sending is canceled. No further sending takes place afterward (`SendAct` = 0).

The ID parameter is the connection number specified in the connection configuration. and is applied only at the first call after a cold restart.

The parameter R_ID is a random number. The R_ID parameter must be identical at the corresponding send and receive blocks. The R_ID parameter is applied only for the first call after a warm restart.

**Note**

**Error code or message at the data block**

When configuring a block pair for data transmission from an SND_AnaVal block to an RCV_AnaVal block, make sure that the settings for ID/R-ID are unique throughout the project.

Only one Rcv block may be assigned to each Snd block or only one connection via ID/R-ID may be configured.

Make sure that you perform the engineering clean and correctly because the data may otherwise be transferred to several blocks or an incorrect block.

### Configuration

The block is installed in the CFC editor in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Time characteristics

The block does not have any time response.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

## 21.1.2 Snd_DigVal modes

This block has no operating mode.

## 21.1.3 Snd_DigVal functions

Configurable reactions using the `Feature` parameter.

An overview of all the reactions that are provided by the `Feature` parameter is available in the section Configurable functions using the Feature I/O (Page 139).

The following functionality is available for this block at the relevant bits:

- Bit 29: Output substitute values if raw value is invalid

### 21.1.4 Snd_DigVal error handling

Error handling of the block is restricted to the error information of the lower-level SFB 12 "BSEND".

The bits of information are set at the `SendErr(ERR)` and `SendStat(STAT)` parameters.

If an error occurs, a new job is automatically triggered with the current data until all data have been successfully transferred.

### 21.1.5 Snd_DigVal messages

No message behavior is applicable to this block.

### 21.1.6 Snd_DigVal I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

#### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `EN` | 1 = Called block will be processed | BOOL | 1 |
| `ID` | ID of physical connection | WORD | 16#0000 |
| `R_ID` | ID of message frame connection | DWORD | 16#00000000 |
| `SndEn` | 1 = Activate continuous sending | BOOL | 1 |
| `PV_In0 ... PV_In127` | Input_0 ... Input_127 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |

#### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `ENO` | 1 = Block algorithm completed without errors | BOOL | 0 |
| `SndAct` | 1 = Command is active | BOOL | 0 |
| `SndDone` | 1 = Command is executed | BOOL | 0 |
| `SndErr` | 1 = Command is executed with error | BOOL | 0 |
| `SndStat` | Type of error | WORD | 16#0000 |

### 21.1.7 Snd_DigVal block diagram

A block diagram is not provided for this block.

# 21.2 Rcv_DigVal - Receive Boolean values including quality code (Rcv_DigVal)

## 21.2.1 Description of Rcv_DigVal

### Object name (type + number) and family

Type + number: FB 1892

Family: COMM

### Area of application for Rcv_DigVal

The "Rcv_DigVal I" block represents a simple user interface to SFB 13 "BRCV". The block receives 128 DigVal structures via MPI, PROFIBUS or Industrial Ethernet connection to another CPU. This CPU must call the function block "Snd_DigVal" (FB 1891) of the PCS 7 Advanced Process Library to send the data.

In STEP 7, a uniform connection must be configured for both communication partners and transferred to the AS.

### How it works

Data are entered in the data block asynchronously to user-program execution. After "`Rcv_DigVal`" has been called, instance DB data may not be processed as long as the job is in progress (`RcvNewData` = 0). If the job is completed without errors, the `RcvNewData` output is set to 1 for one cycle. In the next cycle, the receive enable is automatically sent by the FB to the operating system of the CPU.

The receive enable can take take effect prior to the arrival of the first receive job. In this case, the receive enable is stored by the operating system.

The `ID` parameter is the connection number specified in the connection configuration. and is applied only at the first call after a cold restart.

The parameter `R_ID` is a random number. The `R_ID` parameter must be identical at the corresponding send and receive blocks. The `R_ID` parameter is applied only for the first call after a warm restart.

Call the "Rcv_DigVal" block for the ID/R_ID pair in each program cycle (cyclically or also via timeout alarm). Each message frame requires two calls of "Rcv_DigVal".

**Note**

**Error code or message at the data block**

When configuring a block pair for data transmission from an SND_AnaVal block to an RCV_AnaVal block, make sure that the settings for ID/R-ID are unique throughout the project.

Only one Rcv block may be assigned to each Snd block or only one connection via ID/R-ID may be configured.

Make sure that you perform the engineering clean and correctly because the data may otherwise be transferred to several blocks or an incorrect block.

The outputs `RcvErr` (Error) and `RcvStat` (Status) show the specific error information that correspond to SFB 13 "BRCV".

If an error occurs, use an appropriate substitute strategy with `Feature` (bit 29/30).

If you want to simulate the block, enable simulation via the `SimOn` input. In this case, the values from the inputs `SimPVX` are written to the outputs `PV_Outx`. The data sent are not applied, since data reception is switched off.

### Configuration

The block is installed in the CFC editor in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Time characteristics

The block does not have any time response.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

## 21.2.2 Rcv_DigVal modes

This block has no operating mode.

## 21.2.3 Rcv_DigVal functions

**Configurable reactions using the `Feature` parameter**

An overview of all the reactions that are provided by the `Feature` parameter is available in the section Configurable functions using the Feature I/O (Page 139).

The following functionality is available for this block at the relevant bits:

- Bit 29: Output substitute values in case of communications error (Page 204)
- Bit 30: Output last value in case of communications error (Page 204)

The signal status of all output values with status 16#80 is set to local "Manipulated value" (16#60), all other status values remain unchanged.

## 21.2.4 Rcv_DigVal error handling

Error handling of the "Rcv_DigVal" block is restricted to the error information of the lower-level SFB 13 "BRCV".

You can find additional information in the manual "*System Software for S7-300/400 - System and Standard Functions*".

In the manual "*System Software for S7-300/400 - System and Standard Functions*", you can also find a description of the `RcvErr` and `RcvStat` outputs

The substitute value is output when bit 29 is set in the `Feature` structure and no new data is received after the expiration of "RcvMonCyc" (number of cycles). While the "REC_MON" cycles are running, the last received values are applied to the output.

If Bit29 is not set in the `Feature` structure and an error occurs, the last valid values are always applied to the output.

## 21.2.5 Rcv_DigVal messages

No message behavior is applicable to this block.

## 21.2.6 Rcv_DigVal I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `EN` | 1 = Called block will be processed | BOOL | 1 |
| `ID` | ID of physical connection | WORD | 16#0000 |

| Parameter | Description | Type | Default |
|---|---|---|---|
| R_ID | ID of message frame connection | DWORD | 16#00000000 |
| RcvMonCyc | Number of cycles for receiving monitoring error | INT | 3 |
| SimOn | 1= simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimPV0 ... SimPV127 | Simulation value | BOOL | 0 |
| SubsPV1 ... SubsPV127 | Substitute value | BOOL | 0 |
| Feature | I/O for additional functions (Page 2807)<br>• Reserved<br>• 1 = Output substitute values in case of communications error<br>• 1 = Output last values in case of communications error<br>• Reserved | STRUCT<br>• Bit0 ... Bit28: BOOL<br>• Bit29: BOOL<br>• Bit30: BOOL<br>• Bit31: BOOL | -<br>• 0<br>• 0<br>• 0<br>• 0 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| PV_Out0 ... PVOut127 | Output_00 ... Output_07 | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| SimAct | 1= simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RcvMonErr | 1 = no data received | BOOL | 0 |
| RcvNewData | 1 = no data received | BOOL | 0 |
| RcvErr | 1 = Command is executed with error | BOOl | 0 |
| RcvStat | Type of error | WORD | 16#0000 |

## 21.2.7 Rcv_DigVal block diagram

A block diagram is not provided for this block.

# 21.3 Snd_AnaVal - Send analog values including quality code (SND_AnaVal)

## 21.3.1 Description of Snd_AnaVal

### Object name (type + number) and family

Type + number: FB 1893

Family: COMM

### Area of application for Snd_AnaVal

The "Snd_AnaVal" block represents a simple user interface to SFB 12 "BSEND". The "Snd_AnaVal" block sends 32 analog structures via an MPI, PROFIBUS or Industrial Ethernet connection to another CPU. The other CPU calls the function module "Rcv_AnaVal" (FB 1894) of the PCS 7 Advanced Process Library to receive the data.

### How it works

Data transfer is initiated by calling the block with the value 1 at the `SendEn` control input. The data are sent according to the settings `CylMin`, `CylMax` or `PV_In_HysXX`. While data is being sent, `SendAct` is set to 1. If the job is completed without errors, `SendDone` is set to 1. `SendErr` = 1 if an error occurs. A new job is then automatically sent with the current data until all data have been successfully transferred.

The `ID` parameter is the connection number specified in the connection configuration. and is applied only at the first call after a cold restart.

The parameter `R_ID` is a random number. The `R_ID` parameter must be identical at the corresponding send and receive blocks. The `R_ID` parameter is applied only for the first call after a warm restart.

---

**Note**

**Error code or message at the data block**

When configuring a block pair for data transmission from an SND_AnaVal block to an RCV_AnaVal block, make sure that the settings for ID/R-ID are unique throughout the project.

Only one Rcv block may be assigned to each Snd block or only one connection via ID/R-ID may be configured.

Make sure that you perform the engineering clean and correctly because the data may otherwise be transferred to several blocks or an incorrect block.

---

If input `SendEn` = 0, incomplete sending is canceled. No further sending takes place afterward; `SendAct` = 0.

You can use `CylMin` to set the number of cycles to wait before triggering the transfer of the current input data, regardless of any changes to one or more values.

You can use `CylMax` to specify how many cycles after the last valid data transfer to wait before sending the current input data. Sending is executed even when no value has changed or when the changes of a REAL value are within the hysteresis `PV_In_HysXX`.

The default hysteresis value is `PV_In_HysXX` = 0. If sending is not performed after each change, then configure the `PV_In_HysXX` input accordingly. This involves an absolute value.

When `CylMax` is reached, transfer is forced after the corresponding number of cycles.

### Configuration

The block is installed in the CFC editor in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Time characteristics

The block does not have any time response.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

## 21.3.2 Snd_AnaVal modes

This block has no operating mode.

## 21.3.3 Snd_AnaVal functions

The block has no functions.

## 21.3.4 Snd_AnaVal error handling

Error handling of the block is restricted to the error information of the lower-level SFB 12 "BSEND".

The information is set at the outputs `SendErr`(ERR) and `SendStat`(STAT).

If an error occurs, a new job is automatically triggered with the current data until all data have been successfully transferred.

## 21.3.5 Snd_AnaVal messages

No message behavior is applicable to this block.

## 21.3.6 Snd_AnaVal I/Os

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| EN | 1 = Called block will be processed | BOOL | 1 |
| ID | ID of physical connection | WORD | 16#0000 |
| R_ID | ID of message frame connection | DWORD | 16#00000000 |
| SndEn | 1 = Activate continuous sending | BOOL | 1 |
| CylMin | Minimum waiting cycle | INT | 1 |
| CylMax | Maximum waiting cycle | INT | 10 |
| PV_In_Hys00 ... PV_In_Hys31 | EDC-Hysteresis PV_InR_00 ... EDC-Hysteresis PV_InR_31 | REAL | 0.0 |
| PV_In0 ... PV_In31 | Input_0 ... Input_31 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| SndAct | 1 = Command is active | BOOL | 0 |
| SndDone | 1 = Command is executed | BOOL | 0 |
| SndErr | 1 = Command is executed with error | BOOL | 0 |
| SndStat | Type of error | WORD | 16#00000 |

## 21.3.7 Snd_AnaVal block diagram

A block diagram is not provided for this block.

# 21.4 Rcv_AnaVal - Receive analog values including quality code (Rcv_AnaVal)

## 21.4.1 Description of Rcv_AnaVal

### Object name (type + number) and family

Type + number: FB 1894

Family: COMM

### Area of application for Rcv_AnaVal

The "Rcv_AnaVal" block represents a simple user interface to SFB 12 "BRCV". The block sends 32 analog structures via MPI, PROFIBUS or Industrial Ethernet connection to another CPU. The other CPU calls the function module "Snd_AnaVal" (FB 1893) of the PCS 7 Advanced Process Library to receive the data.

### How it works

Data are entered in the data block asynchronously to user-program execution. After "Rcv_AnaVal" has been called, instance DB data may not be processed as long as the job is in progress (`RcvNewData` = 0). If the job is completed without errors, the `RcvNewData` output is set to 1 for one cycle. In the next cycle, the receive enable is automatically sent by the FB to the operating system of the CPU.

The receive enable can take take effect prior to the arrival of the first receive job. In this case, the receive enable is stored by the operating system.

The `ID` parameter is the connection number specified in the connection configuration. and is applied only at the first call after a cold restart.

The parameter `R_ID` is a random number. The `R_ID` parameter must be identical at the corresponding send and receive blocks. The `R_ID` parameter is applied only for the first call after a warm restart.

Call the "Rcv_AnaVal" block for the `ID/R_ID` pair in each program cycle (cyclically or also via timeout alarm). Each message frame requires two calls of "Rcv_AnaVal".

---

**Note**

**Error code or message at the data block**

When configuring a block pair for data transmission from an SND_AnaVal block to an RCV_AnaVal block, make sure that the settings for ID/R-ID are unique throughout the project.

Only one Rcv block may be assigned to each Snd block or only one connection via ID/R-ID may be configured.

Make sure that you perform the engineering clean and correctly because the data may otherwise be transferred to several blocks or an incorrect block.

---

The outputs `RcvErr` (error) and `RcvStat` (status) show the specific error information that correspond to SFB 13 "BRCV".

If an error occurs, use an appropriate substitute strategy with `Feature.Bit29`.

If you want to simulate the block, enable simulation via the `SimOn` input. In this case, the values from the inputs `SimPVX` are written to the outputs `PV_Outx`. The data sent are not applied, since data reception is switched off.

### Configuration

The block is installed in the CFC editor in a cyclic interrupt OB (OB30 to OB38).

### Startup characteristics

The block does not have any startup characteristics.

### Time characteristics

The block does not have any time response.

### Status word allocation for `Status` parameter

This block does not have the `Status` parameter.

## 21.4.2 Rcv_AnaVal modes

This block has no operating mode.

## 21.4.3 Rcv_AnaVal functions

#### Configurable reactions using the Feature parameter

An overview of all the reactions that are provided by the `Feature` parameter is available in the section Configurable functions using the Feature I/O (Page 139).

The following functionality is available for this block at the relevant bits:

- Bit 29: Output substitute values in case of communications error (Page 204)
- Bit 30: Output last value in case of communications error (Page 204)

The signal status of all output values with status 16#80 is set to local "Manipulated value" (16#60), all other status values remain unchanged.

### 21.4.4 Rcv_AnaVal error handling

Error handling of the "Rcv_AnaVal" block is restricted to the error information of the lower-level SFB 13 "BRCV". You can find additional information in the manual System Software for S7-300/400 - System and Standard Functions. There you can find a description for the `RcvErr` and `RcvStat` outputs.

The substitute value is output when Bit29 is set in the `Feature` structure and no new data is received after the expiration of `RcvMonCyc` (number of cycles). While the `RcvMonCyc` cycles are running, the last received values are applied to the output.

If Bit29 is not set in the `Feature` structure and an error occurs, the last valid values are always applied to the output.

### 21.4.5 Rcv_AnaVal messages

No message behavior is applicable to this block.

### 21.4.6 Rcv_AnaVal I/Os

I/Os of Rcv_AnaVal

For the parameters that are not linkable, refer: Parameters that cannot be connected (Page 75)

#### Input parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| `EN` | 1 = Called block will be processed | BOOL | 1 |
| `ID` | ID of physical connection | WORD | 16#0000 |
| `R_ID` | ID of message frame connection | DWORD | 16#00000000 |
| `RcvMonCyc` | Number of cycles for receiving monitoring error | INT | 3 |
| `SimOn` | 1 = Simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| `SimPV0...SimPV31` | Simulation value | REAL | 0.0 |
| `SubsPV0...SubsPV31` | Substitute value | REAL | 0.0 |
| `Feature` | I/O for additional functions (Page 2813)<br>• Reserved<br>• 1 = Output substitute values in case of communications error<br>• 1 = Output last values in case of communications error<br>• Reserved | STRUCT<br>• Bit0 ... Bit28: BOOL<br>• Bit29: BOOL<br>• Bit30: BOOL<br>• Bit31: BOOL | -<br>• 0<br>• 0<br>• 0<br>• 0 |

### Output parameters

| Parameter | Description | Type | Default |
|---|---|---|---|
| ENO | 1 = Block algorithm completed without errors | BOOL | 0 |
| PV_Out0 ... PV_Out31 | Output_0 ... Output_31 | STRUCT<br>• Value: REAL<br>• ST: BYTE | -<br>• 0.0<br>• 16#80 |
| SimAct | 1 = Simulation active | STRUCT<br>• Value: BOOL<br>• ST: BYTE | -<br>• 0<br>• 16#80 |
| RcvMonErr | 1 = no data received | BOOL | 0 |
| RcvNewData | 1 = new data received | BOOL | 0 |
| RcvErr | 1 = Command is executed with error | BOOL | 0 |
| RcvStat | Type of error | WORD | 16#0000 |

## 21.4.7 Rcv_AnaVal block diagram

A block diagram is not provided for this block.

# Process tag types (insertible templates) 22

## 22.1 Introduction to process tag types

### Introduction

You can find control engineering information on the standard process tag types of the Advanced Process Library in a separate section in this help:

- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 2822)
- PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 2823)
- Step controller with direct access to the actuator and without position feedback (StepControlDirect) (Page 2831)
- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 2832)
- Split-range controller with control loop monitoring through ConPerMon (SplitrangeControl) (Page 2833)
- Ratio control with control loop monitoring through ConPerMon (RatioControl) (Page 2836)
- Ratio control with PIDConR (RatioR) (Page 2839)
- Cascade control with control loop monitoring through ConPerMon (CascadeControl) (Page 2840)
- Cascade control with PIDConR (CascadeR) (Page 2843)
- Cascade control with PIDStepL (CascadeStepControl) (Page 2844)

If you want to do without control loop monitoring in order to reduce license costs or CPU resources, you can use accordingly simplified process tag types, which are identifiable by the suffix "Lean" in their names (for example, PIDControlLean, CascadeControlLean, RatioControlLean) or you can delete the ConPerMon block from the CFC charts for any of the process tag types.

In addition to the process tag types mentioned above, you will also find the following in the Templates folder of the library:

- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 2827)
- PID controller for PA/FF devices (PIDControl_Lean_Fb) (Page 2821)
- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 2825)
- Override control (Page 2846)
- Override control with PIDConR (OverrideR) (Page 2848)
- PID controller with Smith predictor (SmithPredictorControl) (Page 2830)
- Model-based predictive control (ModPreCon) (Page 2849)

This list contains prototype process tag types for several of the complex control loop structures shown in the example project (Page 2870). These prototypes are simply examples of how such process tag types might appear. In most process plants, it is assumed that high-level control loop structures must be configured individually some in combination with lower-level control loops. This means that mass production of high-level control loop structures as instances of process tag type represents the exception.

Descriptions for the following process tag types are also available:

- Monitoring eight digital process tags (Digital8Monitoring) (Page 2853)
- Monitoring of a digital process tag (DigitalMonitoring) (Page 2851)
- Monitoring a digital process tag for PA/FF devices (DigitalMonitoring_Fb) (Page 2852)
- Monitoring an analog process tag (AnalogMonitoring) (Page 2854)
- Monitoring of an analog process tag for PA/FF devices (AnalogMonitoring_Fb) (Page 2855)
- Dosing (Dose_Lean) (Page 2856)
- Dosing with PA/FF devices (Dose_Lean_Fb) (Page 2857)
- Two-speed motor (Motor2Speed) (Page 2860)
- Reversing motor (MotorReversible) (Page 2861)
- Reversible motor with controllable speed (MotorSpeedControlled) (Page 2862)
- Motor (Motor_Lean) (Page 2858)
- Motor with an additional analog value and time-stamped signals (Motor_AV_EventTs) (Page 2863)
- Motor valve (ValveMotor) (Page 2867)
- Valve (Valve_Lean) (Page 2865)
- Two-way valve (Valve2Way) (Page 2866)
- Control valve (VlvAnL)  (Page 2868)
- Control valve for PA/FF devices (ValveAnalog_Fb) (Page 2869)

### Using process tag types

When using process tag types in a PCS 7 multiproject, the following procedure is recommended:

- Copy all the required function blocks from the Advanced Process Library to the block folder of the master data library of the multiproject (<Projectname>_Lib).
- Then in the plant view, copy the required process tag types from the Advanced Process Library to the "Process_tag_types" folder of the master data library of the multiproject.
- Make any customizations of the process tag types.

### Generating the process tags

Process tags are the instances of the process tag types.

There are two ways of generating process tags:

1. Copy the process tag type from the master data library and insert it in the target folder in the plant hierarchy. You can then set the parameters for and interconnect the CFC chart in the CFC Editor.
2. Using the Import-Export assistant, you can then generate the required process tags of the process tag type that you then assign parameters to and interconnect based on an import file.

### Notes on the "Measured value failure" application scenario for controller block process tag types

If there is no valid measured value for `PV`, the controller must not remain in automatic mode because a closed control loop no longer exists. The controller is unable to calculate a useful manipulated variable if there is no feedback of the actual process state. The controller must then be changed to manual or tracking mode. OS operators must be made aware of this situation by a corresponding message. Various reactions to the loss of measured values are conceivable depending on the application background:

1. Tracking a fixed, configured neutral position manipulated variable, for example, Valve closed, Heater off, or similar.
2. Holding the last valid manipulated variable `MV` constant to retain a steady process state as far as possible (if the process was already in such a state).
3. Change to manual mode, so that the OS operator can take over responsibility for process control.

A combination of reactions 2 and 3 is implemented in the template. The controller changes to tracking with the last valid manipulated variable as start value. The switchover is made via the input `MV_TrkOn` of the PID block. Since manual mode already has priority over tracking, the OS operator can subsequently use the command.

Caution: It is not advisable to freeze the last valid manipulated value if the signals are subject to heavy noise, or in control loops with frequent setpoint changes. In such situations, it is advisable to change to variant 1.

If a controller intervenes locally in the process via an actuator (for example, a valve with electropneumatic positioner Sipart PS) specific measures similar to those for a cascade controller must be taken:

- If there is local intervention, the controller tracks the current actuator position. In this case, the feedback signal is applied to input `MV_Trk` and an OR element is set before input `MV_TrkOn`.

### Notes on analog position feedback for controller block process tag types

If there is no position feedback, delete the associated analog input channel block at the `MV_Rbk` parameter in the CFC chart. This will also make the corresponding display elements in the standard view of the faceplate invisible.

# 22.2 PID controller (PIDControl_Lean)

## PID controller

This process tag type forms the basis for generating PID control instances for continuous processes. In addition to the actual PID block, it contains functionality that should be implemented consistently in each control loop. This functionality is, however, not implemented within the PID block itself so that users can adapt the functions to a specific project:

- An analog input channel block for the actual value `PV` plus an analog output channel block for the manipulated variable `MV`.
- Certain logic blocks that change the control loop to a safe mode if the measurement of the process value fails. This state is signaled by the signal status of `PV`.

**Note**

Read the information for process tag types with controller blocks in Introduction to process tag types (Page 2817).

# 22.3 PID controller for PA/FF devices (PIDControl_Lean_Fb)

## PID controller

This process tag type forms the basis for generating PID control instances for continuous processes. In addition to the actual PID block, it contains functionality that should be implemented consistently in each control loop. This functionality is, however, not implemented within the PID block itself so that users can adapt the functions to a specific project:

- An analog input channel block for the actual value `PV` plus an analog output channel block for the manipulated variable `MV`.
- Certain logic blocks that change the control loop to a safe mode if the measurement of the process value fails. This state is signaled by the signal status of `PV`.

**Note**

Read the information for process tag types with controller blocks in Introduction to process tag types (Page 2817).

# 22.4 PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon)

### PID controller with safety logic and control loop monitoring

This process tag type forms the basis for generating PID control instances for continuous processes. In addition to the actual PID block, it contains functionality that should be implemented consistently in each control loop. This functionality is, however, not implemented within the PID block itself so that users can adapt the functions to a specific project:

- An analog input channel block for the actual value `PV` and (if available) the position feedback (`Rbk`), plus an analog output channel block for the manipulated variable `MV`.
- A simple process simulation of the first order, controlled by the manipulated variable `MV` which provides simulation values to analog input `PV`. This functionality allows at least elementary function tests control loop before a real process is connected.
- Certain logic blocks that change the control loop to a safe mode if the measurement of the process value fails. This state is signaled by the signal status of `PV`.
- An additional function block for control loop monitoring that should be installed in all PID control loop.

**Note**

Read also the information on controller block process tag types in Introduction to process tag types (Page 2817).
You can find information on the use of control loop monitoring in ConPerMon functions (Page 845).

# 22.5 PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon)

## PIDConR with safety logic and control loop monitoring

This process tag type corresponds to the process tag type PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 2822) and control loop monitoring when PIDConR is used rather than PIDConL.

# 22.6 PID controller with safety logic and manipulated variable ramp (PIDConR_MV_Ramp)

### PID controller with safety logic and manipulated variable ramp

This process tag type is derived from the process tag type PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 2823). But in this case, a manipulated variable ramp is realized instead of a control loop monitoring.

The degree of manipulated variable is limited outside of PIDConR with the block RateLim. The limited `MV` value is fed back to the input parameter `ExtReset` of the block PIDConR.

# 22.7 PID - control with operating-point-oriented parameter control (GainScheduling)

## PID - control with operating-point-oriented parameter control

Many technical processes have a non-linear response due to non-linear physical, chemical or thermodynamic effects. When such a process needs to be kept in the close vicinity of a fixed operating point, the transfer response can be linearized around this operating point. A linear PID controller can be designed for this linearized transfer function. If, however, the process has a strongly non-linear response and/or operates at different operating points, no constantly good control response can be expected throughout the entire operating range. Due to the non-linearity, various gain factors or process time constants are in effect at different operating points. In keeping with this, different controller parameters will be considered to be optimum.

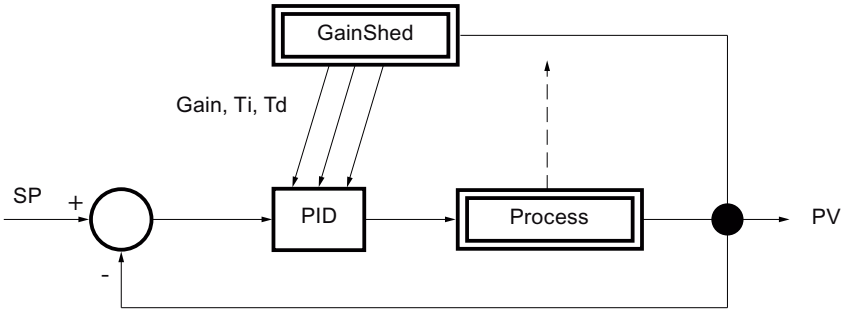



Figure 22-1 Operating-point-oriented parameter adaptation

One possible (the simplest) solution to this problem is known as gain scheduling or parameter scheduling. Using a tool such as the PCS 7 PID Tuner, various experiments are performed at different operating points, in each case with low signal amplitudes. This results in different PID parameter sets for each operating point. Up to three such parameter sets can be stored in the GainSched Function block. The suitable parameter set is selected depending on a continuously measurable variable that describes the state of the process, typically the control variable PV itself. Between the operating points for which there are exact parameter values, the values are calculated by linear interpolation of the neighboring interpolation points so that soft and bumpless transitions are possible between the operating points. The term "parameter scheduling" makes it clear that the "timetable" for adjusting the parameters is specified in advance. In contrast, an adaptive controller adapts itself automatically to the differing process response during operation.

The function block GainSched is produced from the CFC chart "fbGainSched" by compiling it as a block type. This CFC chart is supplied with the library so that the user has the option of

expanding the existing basic functionality as necessary, for example to more than three operating points.

**Note**

The combination of several locally optimized controllers by gain scheduling to form a non-linear controller does not necessarily represent an optimum non-linear controller for the non-linear process when considered from a mathematical point of view. This becomes clear even with benign non-linearities (that are continuous and can be differentiated) when setpoint step changes are made between different operating points. Great caution is needed with non-linearities that are discontinuous or cannot be differentiated or with non-monotonic non-linearities.

## Examples of applications

- Control (especially temperature control) of batch processes, for example, batch reactors and batch columns
- pH value control
- Temperature control with phase transitions (for example, fluid/vapor form)
- Control of semi-batch plants (continuous plants with operating point changes, for example, polymerization reactors)
- Control in power plants with load changes

# 22.8 PID controller with dynamic feedforward control (FfwdDisturbCompensat)

## PID controller with dynamic feedforward control

Feedforward can be used when a known, strong disturbance affects the process and its cause can be measured. In these cases, the following general strategy applies: "Control as much as possible (if known in advance and described by a model), control as much as necessary (the rest including the model error and immeasurable disturbances)".

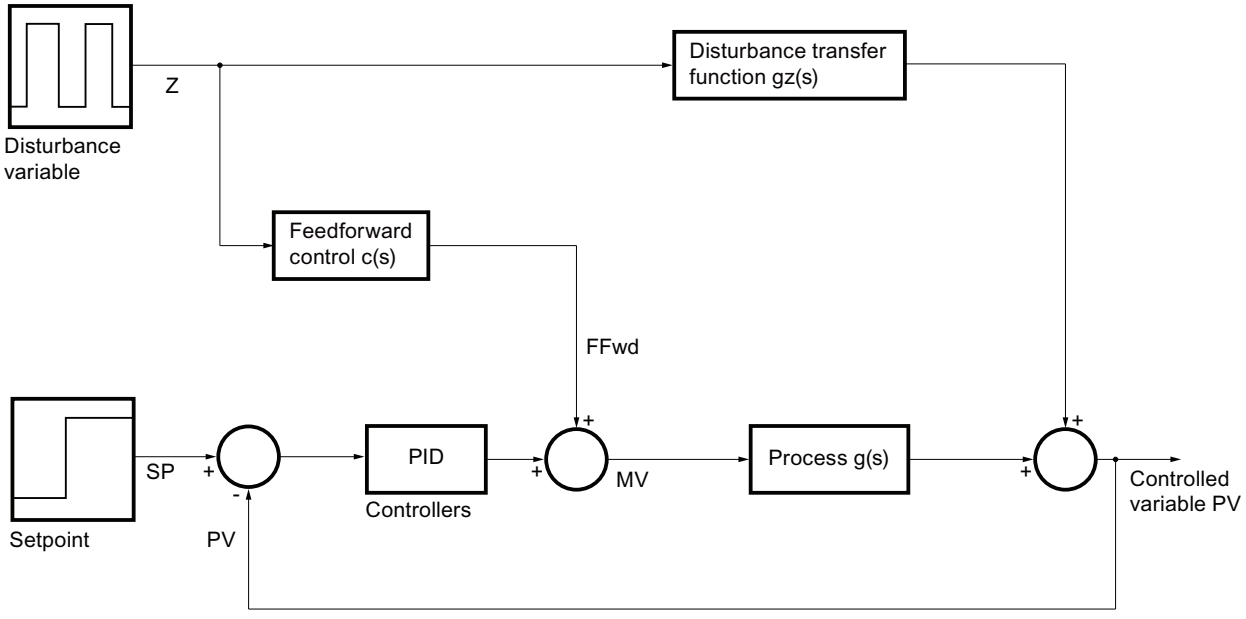



Figure 22-2 Dynamic feedforward control

The effect of a measurable disturbance can be estimated in the form of a transfer function $g_z(s) = y(s) / z(s)$ when the controller is running in manual mode so that no changes whatsoever to the controlled variable y = PV are caused by the manipulated variable and all changes can be attributed to the disturbance z(s).

The transfer function of an ideal feedforward control c(s) can be derived from the requirement that the effect of z on y should be zero for any disturbance signal z(s):

$g_z(s) \cdot z + c(s) \cdot g(s) \cdot z = (g_z(s) \cdot g(s)) \cdot z = 0$

To meet this equation, the compensation block must approximate the equation

$$c(s) = -\frac{g_z(s)}{g(s)}$$

as well as possible. For this to happen, the disturbance transfer function $g_z(s) = y(s) / z(s)$ must be known and the transfer function of the main controlled system $g(s) = y(s) / u(s)$, u = MV must be inverted. If both transfer functions can be modeled as first order with dead time

$$g(s) = \frac{k_s}{1 + t_1 s} \cdot e^{-s\theta}$$

and

$$g(s) = \frac{k_{sz}}{1 + t_{1z} s} \cdot e^{-s\theta_z}$$

and $\theta < \theta_z$ applies, the resulting compensation element must represent the transfer

$$c(s) = -\frac{k_{SZ}}{k_S} \cdot \frac{1 + t_1 s}{1 + t_{1Z} s} \cdot e^{-s(\theta_Z - \theta)}$$

function.

In general, the following dynamic transfer function is required for additive feedforward control:

$$FFwd(s) = -k_c \cdot \frac{t_{cd} s + 1}{t_{cl} + 1} \cdot e^{-\theta_c s} \cdot z(s)$$

where:

$MV = MV_{PID} + FFwd$

In the example above, this function includes the following parameters:

$$k_c = \frac{k_{sz}}{k_s}, \quad t_{cd} = t_1, \quad t_{cl} = t_{1z}, \quad \theta_c = \theta_z - \theta'$$

This transfer function can be created outside the controller with a combination of elementary CFC blocks: one DT1 (Diff),, one PT1 (TimeLag) and one DeadTime block. As shown in the process tag type, the Diff block and TimeLag block are connected in parallel, and the DeadTime block as well as a multiplier are connected in series in front as a gain factor.

The input parameters $k_c$, $t_{cd}$, $t_{c1}$ need to be configured by the user. For a static feedforward control, both time constants are set to zero.

## Examples of applications

- Controlling the outlet temperature of a heat exchanger via steam pressure or heating/cooling medium flow. Flow and inlet temperature of the medium are measurable disturbance variables.
- Fill level control in a drum steam generator using the inlet volume. The outflow is the measurable disturbance variable that is determined by the variable steam consumption in the plant.

- Temperature control in a distillation column using the reflux ration or or heating steam volume. The measurable disturbance variable is the mixture inlet.
- Temperature and concentration control in an agitated tank reactor using cooling medium flow and discharge volume. The temperature and possibly also the concentration of the inflow are measurable disturbance variables.

# 22.9 PID controller with Smith predictor (SmithPredictorControl)

### PID control with Smith predictor

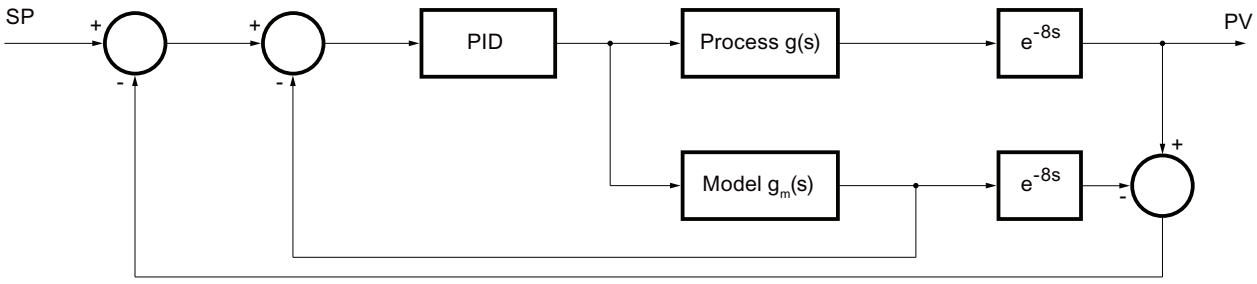



Figure 22-3 PID control with Smith predictor

In processes with large dead times (relative to the dominating time lag constant), a standard PI controller must be set very slowly and compromises must therefore be accepted in the control quality. The control quality can be significantly improved with a Smith predictor that can be derived from the IMC principle (**I**nternal **M**odel **C**ontrol) of model-based control. To achieve this, the transfer function $g_s(s) = g(s) \cdot e^{-s\theta}$ of the controlled system is split up into a part without dead time g(s) and a purely dead time slice $e^{-s\theta}$ with dead time θ. Only the controlled variable y affected by dead time can be measured in the real process. However, a virtual estimate of the controlled variable free of dead time can be taken from the process model (that will become part of the controller) and fed to the controller. This means that the controller itself can be designed for the process without a dead time slice and can therefore be set much more tightly. To compensate unknown disturbances, an estimate of the controlled variable affected by dead time is made in the model and compared with the genuine measured controlled variable. This difference is also fed back to the controller.

In terms of practical application, it must be pointed out that the performance of the Smith predictor depends largely on the model fit, in other words, the dead time must be known. The dead time must be constant or its value must be permanently adapted.

**Note**

To control processes with large dead times, a model predictive controller (see Description of ModPreCon (Page 970)) is also suitable in a single variable situation. It provides greater flexibility in the system modeling and is more convenient thanks to the integrated design procedures, it does however require more CPU resources.

# 22.10 Step controller with direct access to the actuator and without position feedback (StepControlDirect)

### Step controller with direct access to the actuator and without position feedback

The output of the PIDStepL block is directly connected to the process via two digital output channel blocks. This is the simplest form of a step control. The operator can control the actuator (push-button open/close) in the faceplate of the controller. This push-button operation is preferable when operating simple actuators without position feedback since it allows bumpless switchover to auto mode. This switchover is not possible if actuators without position feedback are operated manually from any location outside the control block. For this reason, the template is designed with a PIDStepL block without position feedback.

# 22.11 Step controller with assigned actuator block and position feedback (StepControlActor)

### Step controller with assigned actuator block and position feedback

The output of the PIDStepL block is directly interconnected with the process by way of an actuator block (for example, a motor or a valve). This additional effort is justified when special automation functions of the actuator block, for example motor current monitoring, motor protection and operation in local mode are required. The option of such combinations allows users to do without special functions for operation in local mode or actuator monitoring functions within the controller block.

If the actuator block is not in external default setpoint, or there is an interlock, motor protection or monitoring error, the actuator is incapable of accepting and executing controller commands.

In this context, the resulting structure must be interpreted as a "cascade circuit" consisting of the (primary) controller and actuator block (secondary controller) and measures similar to those for cascade control must be taken. For this purpose, the `CascaCut` output of the actuator block must be interconnected to the `TrkOn` input of the primary controller.

To ensure the bumpless switchover from local or manual mode back to cascade mode, the current actuator position must be fed back to the tracking input of the primary controller. This is, however, only possible when using actuators with position feedback. For this reason, the template is designed for operation with a PIDStepL block with position feedback.

**Note**

The Valve motor valve is configured for operation without motor feedback: `Feature.Bit12` = 1

# 22.12 Split-range controller with control loop monitoring through ConPerMon (SplitrangeControl)

## Split-range controller with control loop monitoring (ConPerMon)

A PID closed-loop controller can use a Split-Range block downstream of the controller output to distribute its manipulated variable to several actuators that influence the same control variable based on different physical principles and in different directions. A typical example of such an application is a temperature control, with heating via a live steam valve and cooling via a cooling water valve. The controller can request heating or cooling energy, depending on the sign of the control deviation (or more precisely of the manipulated variable). In other words, it can work with a bidirectional `MV` output (for example: -100% < `MV` < 100%), although each actuator only supports unipolar operation (for example: 0% < valve position < 100%).

The Split-Range function block contains two separate (static) characteristics for both actuators. Any significant difference between the two actuators in terms of performance (can be interpreted as different steady state gains for heating and cooling), can be compensated by setting different gradients for the characteristics, so that as far as possible, the controller is presented with a linear process response (independent of the sign).

## Example: The cooling effect is not as strong as the heating effect.

If cooling is half as effective as heating, the split-range characteristic for cooling should have twice the gradient.

**Note**

The effective maximum values of the manipulated variables are obtained from the manipulated variable limits of the controller multiplied by the gradients of the split-range characteristics.

The cooling valve cannot go beyond fully open, in other words, the effective limitation for opening the valve is 100%. If a split range characteristic with gradient 2 is used for cooling, the low limit for the manipulated variable must be set to MV_LoLim = -50% on the controller.

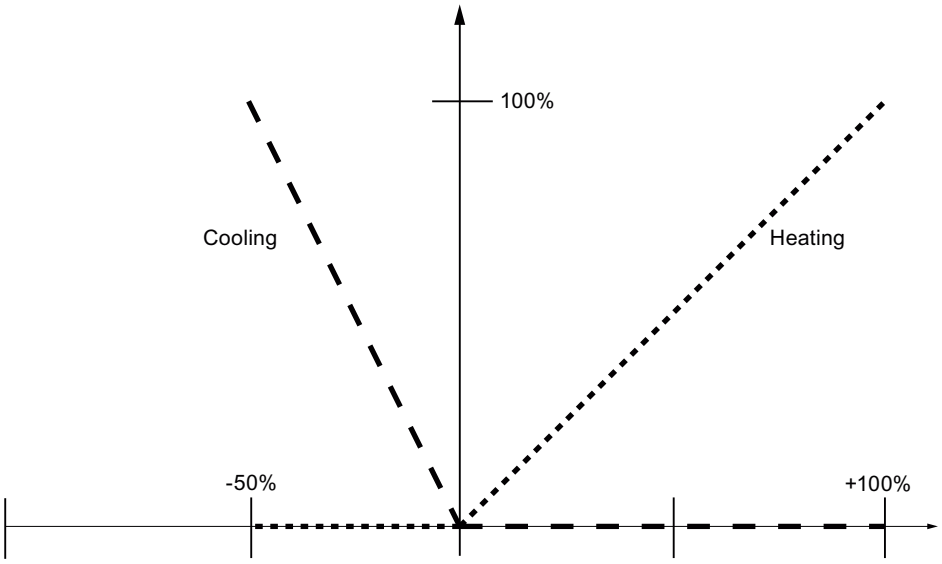



Figure 22-4 Split-range function of the example

The actual split-range function is supplied as separate SplitRange function block that is described in detail in the online help of the block.

**Examples of applications:**

- Control of the temperature of chemical reactors by means of steam heating valves and water cooling valves.
- Temperature control of glass furnaces or sprue channels by means of gas burners and cooling fans
- Temperature control of extruders by means of electrical heating and cooling fans
- Pressure control in a gas phase reactor by means of inlet and outlet valves
- Pressure control at a steam collecting track that supplies steam to several units by means of inlet valves from several steam generators, or the heating power of several steam generators.

# 22.13 Split-Range control (SplitRangeControl_Lean)

### Split-range control

This process tag type is identical to the following process tag type:

Split-range controller with control loop monitoring through ConPerMon (SplitrangeControl) (Page 2833)

The ConPerMon block is not included, however.

# 22.14 Ratio control with control loop monitoring through ConPerMon (RatioControl)

## Ratio control

Specific mixtures of several liquids or gases can be produced by means of a ratio control system consisting of several flow controllers and a Ratio block. The flow setpoint of an additive is obtained from one of these values:

1. From the current `PV` value of the main flow.
   This is the preferred alternative if the primary flow controller operates with steady-state deviation.
   or
2. From the setpoint `SP` of the main flow.
   This alternative provides a smooth, noise-free setpoint signal to the second controller, and allows for a more precise control of specified ratios in transition states where both flow control loops have approximately the same dynamic response.

It is generally advisable to use the second "setpoint-oriented" alternative for main flow controllers with I action.

Interconnect the selected reference value (actual value or setpoint of the main flow) to the In input parameter of the Ratio block.

The ratio control can be expanded by adding further components, in other words, setpoints 3 to n can be derived from `SP1` (or `PV1`) with the help of additional Ratio blocks.

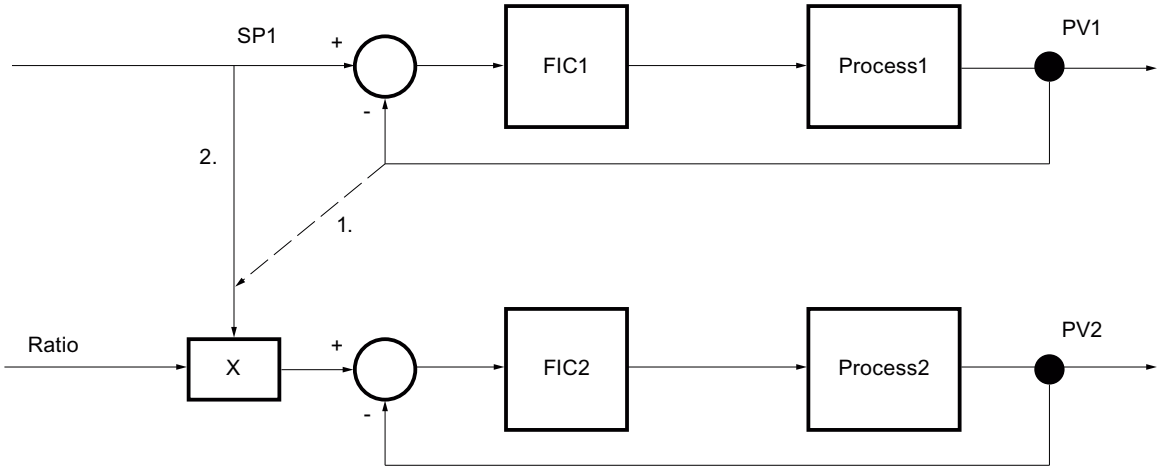



Figure 22-5 Ratio control, process value-oriented (1.) and setpoint-oriented (2.)

The main task of the Ratio block is to define the external setpoint of the secondary (added) component in accordance with

$$Out = In \cdot Ratio + Offset$$

. The actual ratio of the two flow actual values is also calculated in accordance with

$$RatioPV = \frac{SecComPV - Offset}{InPV}$$

and displayed on the control panel for checking purposes. In addition, interconnect the actual value of the main flow to the input parameter `InPV` of the Ratio block and the actual value of the flow of the secondary component to the input parameter `SecComPV`.

In both cases 1 and 2, the current value of the ratio is not controlled directly in the closed loop (no feedback-control) but rather uses feedforward-control.

**Note**

If the PID_Componet1 secondary controller is switched from automatic to another operating mode, the `CascaCut` output parameter is set and the Ratio block is switched to external. In this case, the external setpoint is corrected to the actual value ratio for the ratio of the control actual value to the component actual value: RatioExt = RatioPV. This results in a soft transition to automatic mode.

An additional function block for control loop monitoring must be installed, because it should always be installed in every PID control loop.

## 22.15 Ratio control (RatioControl_Lean)

### Ratio control

This process tag type is identical to the following process tag type:

Ratio control with control loop monitoring through ConPerMon (RatioControl) (Page 2836)

The ConPerMon block is not included, however.

# 22.16 Ratio control with PIDConR (RatioR)

### Ratio control with PIDConR

This process tag type corresponds to the process tag type Ratio control with control loop monitoring through ConPerMon (RatioControl) (Page 2836) when PIDConR is used rather than PIDConL.

You need to set `Feature.Bit21` Switching operator controls for external setpoint to visible (Page 154) to 1 with a flow controller for the secondary component.

# 22.17 Cascade control with control loop monitoring through ConPerMon (CascadeControl)

## General information on cascade control

A cascade control involves two or more PID controllers connected in series. The manipulated variable of the primary controller is interconnected to the external setpoint of the secondary controller so that both control loops are nested. The advantage of cascade control is that disturbances affecting the inner loop can be compensated much more quickly in the secondary loop than in the slower primary loop. In some situations, non-linear effects of the actuator can be compensated in the secondary loop so that a linear process response can be created for the primary loop. Cascade control is possible only when there are further measurable variables in the process in addition to the main control variable and when the internal control loop is significantly faster than the external loop.

## Aspects to be clarified for the cascade control

With any cascade control, the following aspects must be carefully considered and clarified:

- The actuation range of the primary controller must match the setpoint range of the secondary controller to ensure proper operation of the anti-Windup functions of the primary controller.
- If the secondary controller is not operated in "cascade" mode (automatic mode with external setpoint) but in any other mode (for example manual or automatic mode with local setpoint) and therefore does not respond to commands of the primary controller, the primary controller must be put into "tracking" mode to prevent integration of the I action in the primary controller. The manipulated value of the primary controller tracks the process value or setpoint of the secondary controller to allow a bumpless return to cascade mode. The difference between tracking the setpoint and tracking the process value becomes apparent when the secondary controller is put into manual mode. If the process value is tracked, the response is similar to the "Track setpoint to process value in manual mode" of a simple controller.
- If the secondary controller reaches a (high/low) manipulated variable limit, the integrator of the primary controller should be blocked to prevent it going any further in this direction (up / down). The secondary controller cannot go any further in this direction anyway. This prevents any windup of the primary controller when the real actuator has already reached its physical limits while the primary controller has not yet reached its manipulated variable limits.
  Caution:
  - Swap the two bits of this interconnection if the secondary controller has a negative gain.
  - If the secondary controller then reaches a high or low limit, the integrator of the primary controller must be prevented from further integration in this direction.

### Procedure

When you set up and commission the controller, you work from the "inside to the outside", in other words, you start by making the settings for the secondary controller and putting it into auto mode. You then set the primary controller parameters and change the secondary controller to cascade mode. When you set the parameters for the primary controller, remember that from its perspective the entire, inner closed control loop is the "controlled system". The parameters you set for the primary controller depend to a greater or lesser extent on the settings you made for the secondary controller. This becomes less important the greater the difference in dynamic response between the primary and secondary control loop.

### Priorities

The tracking by the primary controller that is initiated by the secondary controller has lower priority than the manual mode on the primary controller. In turn, manual mode has lower priority than forced mode on the master controller as can, for example, be requested by external logic during an emergency shutdown of the plant (controller input `MV_Forced`, unlimited, activated by `MV_ForOn` with highest priority). For this reason, the PIDConL block for cascade control has an additional tracking input `MV_Trk` that is activated by `MV_TrkOn` and is subject to the normal manipulated variable limiting and has lower priority than manual mode.

### Additional application examples

- Temperature control of a distillation column (primary controller) based on the reflux ratio (secondary controller at the top of the column), and heating steam flow rate (secondary controller at the column sump),
- Temperature control of a furnace using a secondary controller to control the fuel flow rate,
- Fill level control for a container using a secondary controller for the inlet and/or outlet flow.
- Position control (in drive engineering) with a secondary controller for the speed and torque.

### Using secondary flow controllers

Secondary controllers are usually used for flow control to prevent detrimental effects on the primary controller resulting from changes in flow rate. The non-linearities frequently often found in a flow control element (for example a valve) are "hidden" in the secondary loop because the secondary closed loop has a linear response and non-linearities have no effect on the primary controller or its tuning.

**Note**

Control loop monitoring is only practical for primary controllers, as explained in the description of the ConPerMon block in the section, Functions/Cascade Control.

# 22.18 Cascade control (CascadeControl_Lean)

## Cascade control

This process tag type is identical to the following process tag type:

Cascade control with control loop monitoring through ConPerMon (CascadeControl) (Page 2840)

The ConPerMon block is not included, however.

# 22.19 Cascade control with PIDConR (CascadeR)

### Cascade control with PIDConR

This process tag type corresponds to a large extent to the process tag type Cascade control with control loop monitoring through ConPerMon (CascadeControl) (Page 2840) when PIDConR is used rather than PIDConL.

Special note: The current `PV_Out` output of the PID_Slave secondary controller is used as an external Reset `ExtReset` on the primary controller. Unlike cascade control, the PID_Master primary controller does not therefore have to be changed to tracking when the cascade is disconnected with all other PID controllers. There is also no need for the direction-dependent blocking of the integrator for the primary controller.

Since the primary controller is reliant on the external Reset, any control deviations in the secondary closed loop interfere with the primary controller; secondary controllers without I action are not therefore recommended for PIDConR.

You need to set the `Feature` bit Switching operator controls for external setpoint to visible (Page 154) to 1 with a secondary controller of the cascade.

# 22.20 Cascade control with PIDStepL (CascadeStepControl)

### Cascade control with a step controller as a valve positioner and a downstream actuator block

The step controller is used in this template as a positioner, which regulates the position of a valve that does not have an internal positioner and is only controlled via binary inputs (open, stop, close). The motorized control valve can thus include delay behavior, i.e. that does not stop immediately at a stop command. The output of the PIDStepL block is directly connected to the process via the actuator block VlvMotL (see StepControlActor template). The PIDStepL runs as a slave controller in a cascade structure and gets his external setpoint from a master controller (flow or pressure regulator).

The PIDStepL is configured as a step controller without position feedback (WithRbk =0). The position feedback signal is applied in this example to the `PV` input and controlled to the external setpoint, which is also set by higher-level master controller (PIDConL). Master and slave controllers are connected in such a way that the following conditions are met (see CascadeControl template):

- The calculated valve position of the master controller is forwarded to the external setpoint of the slave controller.
- The master controller is switched to tracking if there is a bad process value on the master controller or with an open loop by the PIDStepL slave controller or by the VlvMotL valve block. The tracking value corresponds to the valve position at the `PV` input of the PIDStepL step controller. In case of an open loop by the VlvMotL valve block, the open and close control signals of the PIDStepL slave controller are also tracked.
- If the valve position is fully open or fully closed, the I-component of the master controller is stopped (Windup measure).

The same rules as those set in the CascadeControl template and described in the Procedure section apply for the controller setting and commissioning.

# 22.21 Source chart for GainSched function block (gain scheduling)

## Source chart for GainSched function block

In contrast to all other function blocks, the `GainSched` block is implemented as a CFC chart and is generated with the "Compile chart as block type" function. Several application options are available in this case:

- You can use the precompiled function block `GainSched` from the library if the standard functionality is adequate for your needs.
- If you require special additional functions for gain scheduling in your application (for example more than three operating points, additional logic functions for selecting the parameters), you will need to modify the CFC source chart and compile it as a block type with a different FB number.

Internally, the `GainSched` block consists essentially of three instances of the Polygon block, one for each of the three controller parameters, Gain, TI and TD.

The polygon block itself would extrapolate linearly outside of the boundary points and thereby exceed the value range of its interpolation points. However, as this presents too high a risk for controller parameters, the controller parameters which are output are effectively limited to the table values at the boundary operating points by the automatic introduction of additional interpolation points with a horizontal end tangent outside of the specified range.

# 22.22 Override control

## Override control

In an override control, two or more controllers share a common actuator. Depending on the current process state, a decision is made as to which controller actually has access to the actuator, in other words, the various controllers can override each other.

A typical use case is a gas pipeline with pressure and flow control using a single valve. The main aim of the control is to achieve a certain flow rate, however due to safety considerations, the pressure must be kept within certain limits. The pressure controller is therefore known as the "limiting controller" or "secondary controller".

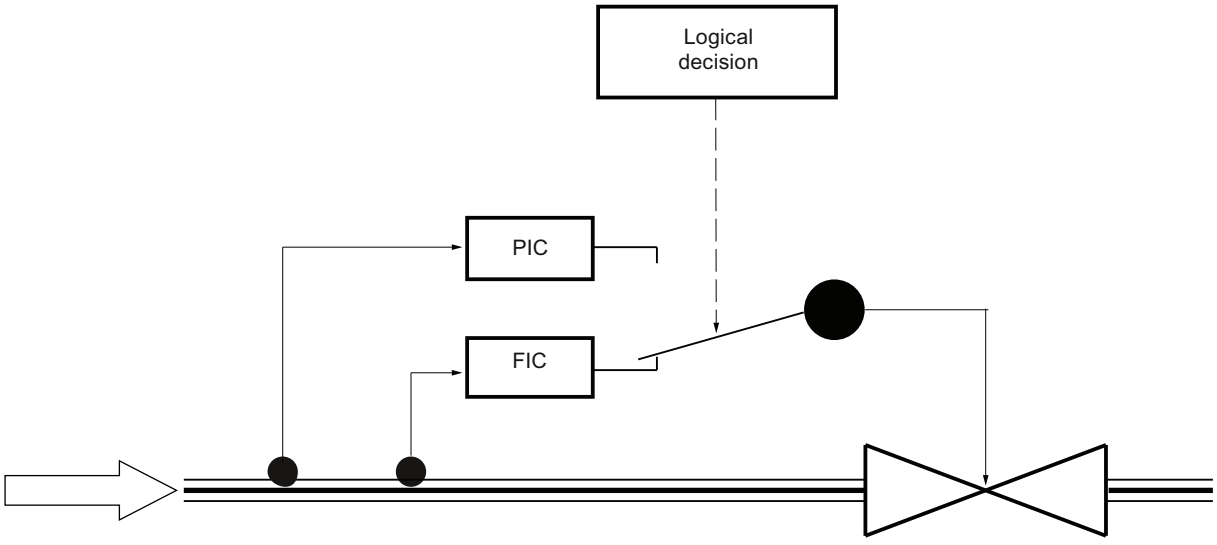



Figure 22-6 Override control with main controller FIC and limiting controller PIC

### Criteria for the type of override control

The logical decision as to which controller should be active can be made based on two different criteria resulting in two different types of override controls:

1. The decision is based on a measurable process output variable, for example one of the two controlled variables. In the example above, the warning limits of the pressure controller can be used to decide whether the pressure controller should be active. The passive controller is in tracking mode to avoid Windup problems and to ensure bumpless transfer. The setpoint of the secondary controller must be somewhat lower than the switchover threshold so that the transfer can be reversed again. This type of override control is easy to understand and to implement. Its advantage is that the high and low limit of the secondary controlled variable (for example pressure) can be monitored; its disadvantage is that a limit cycle oscillation results as soon as the limiting controller needs to intervene. The secondary controller will always attempt to return its controlled variable to the safe range and to return command to the main controller (for example flow rate) so that the active and passive controllers swap over continuously. This variant is therefore only recommended when the secondary controller is seldom required and functions mainly as a safety or backup system.
2. The decision is based on a comparison of the manipulated variables of both controllers, for example the controller that demands the higher (or lower) controlled variable takes control of the actuator. In the example above, the controller that wants to open the valve further takes over control. The setpoint of the secondary controller defines the switching threshold. Both controllers run the entire time in automatic mode. To avoid Windup problems, the manipulated variable limits must be tracked in a crossover structure: When the higher (lower) manipulated variable wins, the low (high) limits of all controllers of the currently highest (lowest) manipulated variable must be corrected slightly up or down by, for example, 2% of the manipulated variable range. This means that this scheme can also be used in applications with more than two controlled variables. There is no Windup problem at the high limit because the highest manipulated variable takes over control anyway. This approach avoids the limit cycle oscillation of alternative 1 but is, in principle asymmetrical, in other words either a high or a low limit of the secondary controlled variable can be monitored but not both.
   This type of override control is described in most control textbooks, particularly in the USA. It can, however, only be used with PID algorithms that allow online manipulation of the manipulated variable limits (in PCS 7 as of V6.0).

### Additional application examples

- **Steam generator:** The primary controlled variable is the steam pressure but the water level in the steam tank must be monitored so that the heating coils remain completely covered by water and the tank does not overflow. The only manipulated variable is the outlet valve.
- **Compressor:** The primary controlled variable is the throughput but the pressure must be monitored to make sure it does not exceed a safety limit. The only manipulated variable is the motor speed.
- **Steam distribution system:** Every plant involving industrial processes has a network of pipes to distribute steam at various pressures throughout the plant. The high pressure of the steam is reduced to lower levels via a valve. The primary controlled variable is the pressure at the lower-level stage, however the pressure in the high pressure piping must also be monitored to make sure that it does not exceed a safety limit.

# 22.23 Override control with PIDConR (OverrideR)

### Override control with PIDConR

This process tag type corresponds for the most part to the process tag type Override control (Page 2846) when PIDConR is used rather than PIDConL.

Special note: The current manipulated variable output at the final controlling element (e.g. the maximum `MaxMV.Out` manipulated variable proposed by the main controller and limiting controller) is used as the external Reset`ExtReset` for both controllers. Unlike override control, the manipulated variable limits (e.g. `MV_LoLim`) of both controllers do not have to be tracked within a defined space (e.g. `MaxMV_Minus2.Out`) by the manipulated value output at the final controlling element with all other PID controllers.

# 22.24 Model-based predictive control (ModPreCon)

## Model-based predictive control

The process tag type shows how users can expand the ModPreCon block by adding extra functions:

- External alarming with the MonAnL block
- Control quality monitoring with the ConPerMon block
- Safety logic for measure value loss.

You can find details about the ModPreCon block in Description of ModPreCon (Page 970).

## Note on using the ConPerMon block in a system with multi-variable

The mathematical concept of the ConPerMon block is designed for single variable applications. If any increase of variance is detected in the channel of a multi-variable control, the ConPerMon algorithm is unable to determine whether this problem is caused by its own internal control channel or by interaction of neighboring channels. It may be helpful, however, to include a ConPerMon block for each control channel of a multi-variable system to monitor whether the control performance during operation remains within the range determined during commissioning. To achieve this, several logic operations must be performed before the `ManSuprCPI` input bit of each ConPerMon block:

- If one or several of the other channels of the multi-variable system are are not in a steady state ("root caused in this channel") due to local causes (for example a setpoint step change) that is indicated by output bit `CPI_SuRoot = 1`, the increase of variance in its own channel cannot be avoided and should not trigger a CPI warning in its own channel.
- If one or several other channels of the multi-variable system exhibit strong variations as indicated by output bit `CPI_WrnAct = 1`, the increase in variance in its own channel cannot be avoided and should not trigger a CPI warning in its own channel. This may be helpful to localize the actual cause of the problem. The channel in which excessive variations are first detected outputs the first alarm; the other channels which may only be affected by errors resulting from the first error do not generate their own alarms.

## Examples of applications

- Quality control in distillation columns, for example control of the column top and sump temperature based on the reflux ratio and heating steam volume
- Temperature control of several adjacent zones of furnaces with several burners, for example, tunnel furnaces, glass smelting plants, glass sprue channels etc.
- Quality control in chemical reactors by adjusting of reaction conditions such as pressure, temperature, feed/drain etc.

- Vaporizers, for example, drum steam generators
- Mills, for example, cement mills, sieve mills: quality control (grain size) in combination with flow rate maximized, manipulated variables: sieve speed and mill feed

**Note**

The statistical evaluation of the service factor (time slice in automatic mode) and the time slices in the manipulated variable limits can be performed in a WinCC trend control, as in the customary case of a single variable. The binary variables from ModPreCon required for this are archived automatically. However, an appropriate trend control must be manually configured on the operator station, since the view archive in the ConPerMon block is not designed for such quantities.

# 22.25 Monitoring of a digital process tag (DigitalMonitoring)

### Monitoring a digital process tag

This process tag type serves as a basis for monitoring a digital process tag using the MonDiL block.

The digital measurement signal is read from the I/O through the PCS7DiIn block.

The process tag type contains the required interconnections between Pcs7DiIn and MonDiL.

# 22.26 Monitoring a digital process tag for PA/FF devices (DigitalMonitoring_Fb)

## Monitoring a digital process tag

This process tag type serves as a basis for monitoring a digital process tag using the MonDiL block.

The digital measurement signal is read from the I/O through the FbDiIn block.

The process tag type contains the required interconnections between FbDiIn and MonDiL.

# 22.27 Monitoring eight digital process tags (Digital8Monitoring)

## Monitoring eight digital process tags

The process tag type serves as a basis for monitoring up to eight digital process tags using the MonDi08 block.

The digital measurement signals are read from the I/O through the PCS7DiIn block.

The process tag type contains the required interconnections between the eight Pcs7DiIn and MonDi08 blocks.

# 22.28 Monitoring an analog process tag (AnalogMonitoring)

## Monitoring an analog process tag

This process tag type serves as a basis for monitoring an analog process tag using the MonAnL block.

The analog value is read from the I/O through the PCS7AnIn block.

The process tag type contains the required interconnections between Pcs7AnIn and MonAnL.

If you wish to monitor and forward the slope of the process value, you usually need to smooth the process value. You can insert a filter block such as Smooth between Pcs7AnIn and MonAnL for this purpose. The gradient of the process value can also be smoothed using the TimeLag parameter at MonAnL.

Procedure for smoothing the process value and its gradient:

1. Check the process value PV in the trend recorder and smooth it using a filter block such as Smooth only as much as is needed.

2. Check the gradient PV_Grad of the process value in the trend recorder and smooth the trend at the TimeLag parameter of MonAnL.

The sum of the time constants of Timelag from MonAnL and TimeConstant from Smooth approximately result in the length of a varying time window in which the gradient is averaged.

# 22.29 Monitoring of an analog process tag for PA/FF devices (AnalogMonitoring_Fb)

## Monitoring an analog process tag

This process tag type serves as a basis for monitoring an analog process tag using the MonAnL block.

The analog value is read from the I/O through the FbAnIn block.

The process tag type contains the required interconnections between FbAnIn and MonAnL.

If you wish to monitor and forward the slope of the process value, you usually need to smooth the process value. You can insert a filter block such as Smooth between FbAnIn and MonAnL for this purpose. The gradient of the process value can also be smoothed using the TimeLag parameter at MonAnL.

Procedure for smoothing the process value and its gradient:

1. Check the process value PV in the trend recorder and smooth it using a filter block such as Smooth only as much as is needed.

2. Check the gradient PV_Grad of the process value in the trend recorder and smooth the trend at the TimeLag parameter of MonAnL.

The sum of the time constants of Timelag from MonAnL and TimeConstant from Smooth approximately result in the length of a varying time window in which the gradient is averaged.

# 22.30 Dosing (Dose_Lean)

## Dosing

This process tag type serves as a basis for dosing either as "Single component dosing via flow measurement" or as "Fill/extraction weighing via dosing scale" using the DoseL block.

The analog measurement signal is read from the I/O through the PCS7AnIn block.

The interlock signals of DoseL are interconnected to the Intlk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The digital output signals are sent to the I/O through the PCS7DiOu blocks.

The process tag type contains the required interconnections between the blocks mentioned above.

# 22.31 Dosing with PA/FF devices (Dose_Lean_Fb)

## Dosing

This process tag type serves as a basis for dosing either as "Single component dosing via flow measurement" or as "Fill/extraction weighing via dosing scale" using the DoseL block.

The analog measurement signal is read from the I/O through the FbAnIn block.

The interlock signals of DoseL are interconnected to the Intlk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via FbDiIn.

The digital output signals are sent to the I/O through the FbDiOu blocks.

The process tag type contains the required interconnections between the blocks mentioned above.

# 22.32 Motor (Motor_Lean)

## Motor with current monitoring

This process tag type serves as a basis for controlling motors with one control signal using the MotL block.

The feedback signal of the motor is read from the I/O through the PCS7DiIn block.

The interlock signals of MotL are interconnected to the IntLk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The digital output signals are sent to the I/O through the PCS7DiOu blocks.

The process tag type contains the required interconnections between the blocks mentioned above.

# 22.33 Motor with PROFIdrive Drive Profile telegram 1 and 20 (Namur)

### Motor with adjustable speed and two directions of rotations according to the speed control mode with standard telegram 1 and 20

The process tag type serves as a basis for control of motors according to the profile "Speed control mode with standard telegram 1 and 20 with and without Namur" with the help of blocks FbDrive and MotSpdCL.

The interlock signals of MotSpdCL are connected to the IntLk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The input/output signals of MotSpdCL are sent to the I/O through the FbDrive block.

The process tag type contains the required interconnections between the blocks mentioned above.

# 22.34 Two-speed motor (Motor2Speed)

### Two-speed motor

This process tag type serves as a basis for controlling motors with two speeds using the MotSpdL block.

The speed feedback signals of the motor are read from the I/O through the PCS7DiIn blocks.

The interlock signals of MotSpdL are connected to the IntLk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The digital output signals are sent to the I/O through the PCS7DiOu blocks.

The process tag type contains the required interconnections between the blocks mentioned above.

# 22.35 Reversing motor (MotorReversible)

### Reversing motor

This process tag type serves as a basis for controlling reversing motors using the MotRevL block.

The feedback signals of the motor are read from the I/O through the PCS7DiIn blocks.

The interlock signals of MotRevL are connected to the IntLk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The digital output signals are sent to the I/O through the PCS7DiOu blocks.

The process tag type contains the required interconnections between the blocks mentioned above.

# 22.36 Reversible motor with controllable speed (MotorSpeedControlled)

### Reversible motor with controllable speed

This process tag type serves as a basis for controlling reversible motors with controllable speed using the MotSpdCL block.

The digital feedback signals of the motor are read from the I/O through the PCS7DiIn blocks.

The speed feedback of the motor is read from the I/O through the PCS7AnIn block.

The interlock signals of MotSpdCL are connected to the IntLk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The digital output signals are sent to the I/O through the PCS7DiOu blocks.

The speed control signal is output to the I/O through the PCS7AnOu block.

The process tag type contains the required interconnections between the blocks mentioned above.

# 22.37 Motor with an additional analog value and time-stamped signals (Motor_AV_EventTs)

## Motor with an additional analog value and time-stamped signals

This process tag type shows the monitoring of an additional analog value, for example, for motor current monitoring (AV block) and additional binary signals (EventTs block) at a technologic block with MotL as an example.

By interconnecting the AV or EventTs block, messages of this block are displayed in the message view of the technologic block connected to it and can be acknowledged there.

## Use of the AV block

The analog signal to be monitored can be read via the Pcs7AnIn block, for example. This has to be interconnected to the AV block. The `AV_Tech` output parameter of the AV block has to be interconnected to the `AV` input parameter of the technologic block.

## Use of the EventTs block

A non-time-stamped binary signal from the I/O can be read via the Pcs7DiIn block, for example. The `PV_Out` output parameter of Pcs7DiIn has to be interconnected to the `Inx` input parameter of the EventTs block. The time stamp of this signal is generated by the EventTs block at signal change.

A time-stamped binary signal from the I/O must be read via the Pcs7DiIT block. The `TS_Out` output parameter of Pcs7DiIT has to be interconnected to the `InTSx` input parameter of the EventTs block.

The `EventTsOut` output parameter of the EventTs block has to be interconnected to the EventTsIn input parameter of the technologic block.

## Additional interconnections

The interlock signals of MotL are interconnected to the IntLk02 interlock blocks. In turn, these interlock blocks are interconnected to other blocks, for example, to digital process tags via the Pcs7DiIn block.

The digital output signal is output to the I/O through the PCS7DiOu block.

The process tag type contains the required interconnections between the blocks mentioned above.

# 22.38 Motor according to the profile for low voltage switchgear devices with profile 1 of the MM_Starter

### Reversible motor according to the manage motor starter with profile type 1

The process tag type serves as a basis for control of motors according to the profile "Manage motor starter - profile type 1" with the help of blocks FbSwtMMS and MotRevL.

The interlock signals of MotRevL are connected to the IntLk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The input/output signals of MotRevL are sent to the I/O through the FbSwrtMMS block.

The process tag type contains the required interconnections between the blocks mentioned above.

# 22.39 Valve (Valve_Lean)

## Valve

This process tag type serves as a basis for controlling a valve with two positions (open/close) using the VlvL block.

The feedback signals of the valve are read from the I/O through the PCS7DiIn blocks.

The interlock signals of VlvL are connected to the IntLk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The digital output signals are sent to the I/O through the PCS7DiOu blocks.

The process tag type contains the required interconnections between the blocks mentioned above.

# 22.40 Two-way valve (Valve2Way)

### Two-way valve

This serves as a basis for controlling a

- multi-way valves with up to three switching positions or
- three individual valves (valve network) to implement a 2-way valve circuit with neutral position

using the Vlv2WayL block.

The feedback signals of the valve or valves are read from the I/O through the PCS7DiIn blocks.

The interlock signals of Vlv2WayL are connected to the IntLk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The digital output signals are sent to the I/O through the PCS7DiOu blocks.

The process tag type contains the required interconnections between the blocks mentioned above.

# 22.41 Motor valve (ValveMotor)

## Motor valve

This process tag type serves as a basis for controlling a motor valve using the VlvMotL block.

The feedback signals of the valve are read from the I/O through the PCS7DiIn blocks.

The interlock signals of VlvMotL are connected to the IntLk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The digital output signals are sent to the I/O through the PCS7DiOu blocks.

The process tag type contains the required interconnections between the blocks mentioned above.

# 22.42 Control valve (VlvAnL)

### Analog control valve

This process tag type serves as a basis for controlling an analog control valve (0 to 100%) using the `VlvAnL` block.

The feedback signals of the valve's total travelled positions (open/closed) are read by the I/Os through the `Pcs7DiIn` blocks.

The feedback signal of the current analog position is read through the `Pcs7AnIn` block.

The interlock signals of `VlvAnL` are interconnected to the `IntLk02` interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via `Pcs7DiIn`.

The analog actuating signal is output to the peripherals via the `Pcs7AnOu` blocks.

The process tag type contains the required interconnections between the blocks mentioned above.

# 22.43 Control valve for PA/FF devices (ValveAnalog_Fb)

### Analog control valve

This process tag type serves as a basis for controlling an analog control valve (0 to 100%) using the `VlvAnL` block.

The feedback signals of the valve's total travelled positions (open/closed) and the current analog position of the value are read by the I/Os through the `FbAnOu` blocks.

The analog actuating signal is output to the peripherals via the `FbAnOu` blocks.

The process tag type contains the required interconnections between the blocks mentioned above.

The interlock signals of `VlvAnL` are interconnected to the `Intlk02` interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via `FbDiIn`.

# 22.44 Example project APL_Example_xx

## 22.44.1 Introduction to the PCS 7 example project for Advanced Process Control

### Introduction

This document relates to the PCS 7 example project, Process Control (APL_Example_xx, xx indicates the language variant) for the PCS 7 Advanced Process Library.

In contrast to the process tag types (insertable templates), the example project is primarily intended for training purposes:

The main aim was to familiarize users with with the new advanced process control structures by allowing them to experiment without having to intervene in the real process. The examples provide realistic process simulation. Working with these examples helps you to understand the concept and specific structural requirements, and to assess the uses of these functions before you implement them in a real system. For this reason, the examples contain a third-order simulation model with gain, equivalence value, and measurement noise but no analog channel blocks. The process model is supplied as the "ProcSimC" CFC chart and incorporated in the examples using the chart-in-chart technique.

The following examples are provided:

- Cascade control of temperature by using the heat flow (CascadeSim) (Page 2873)
- Control loop monitoring for simulation with colored noise (ConPerMonSim) (Page 2875)
- Feedforward control to compensate a measurable disturbance variable (DisturbCompSim) (Page 2875)
- Operating point-oriented adaptation of parameters (gain scheduling) for non-linear processes (GainSchedSim) (Page 2876)
- Override control on a pipeline (OverrideSim) (Page 2877)
- Smith predictor for a dead time system (SmithPredictorSim) (Page 2877)
- Filtering of noisy measured values in a control loop (SigSmoothSim) (Page 2877)
- Predictive control of a 2x2 multi-variable controlled system (ModPreConSim) (Page 2879)
- Predictive control of a non-linear process (ModPreConNonLinSim) (Page 2879)

A template for Fuzzy Control using the *PCS 7 add-on product FuzzyControl++* can be downloaded from the Internet pages of *Siemens I&S* and is therefore not included in the demo project.

A separate tree folder ("Unit") is provided for each example in the PCS 7 example project. Each folder contains a CFC chart with an interconnection example, a brief explanatory text, and an assigned OS picture with self-explanatory visualization of the process example based on a pre-configured trend recorder. A short text in the OS picture describes commissioning and presentation.

## 22.44.2 Process simulation including noise generator (ProcSimC; ProcSimS)

### Process simulation including noise generator

Users require only a few standard blocks to create a dynamic process model that reflects the response pattern of many technological processes with adequate precision. This model is used in all example projects (APL_Example_xx). However, you can also use this model for sales presentations or to test closed-loop control functions, in other words in a project phase in which the real plant is not yet available ("virtual process", "shadow plant"). Process simulation is supplied as an open source CFC chart "ProcSimC" that users can install in other CFC charts as a nested chart (chart-in-chart ). It contains three first order delay elements, a gain factor, an equivalence value `PV` (for `MV = 0`), and a noise generator for white measurement noise. An additive input is provided for (artificial) interference of the input. The Laplace transformation function described below is implemented by means of this model.

$$PV(s) = \frac{Gain}{(TimeLag1 \cdot s + 1)(TimeLag2 \cdot s + 1)(TimeLag3 \cdot s + 1)}(MV(s) + DisV(s)) + PV_0 + Noise$$

### Use cases

Users can adapt this flexible model to suit the requirements of various use cases, for example:

- **Simulation of temperature control systems:** `PV0` represents the temperature without heating, for example, the ambient temperature. The value of `TimeLag1` is typically significantly higher than `TimeLag2` and `TimeLag3`. The value of the latter can also be zero. The sensor develops a typical quantization noise of 0.1° C. The gain is positive can be interpreted as the theoretical maximum temperature that can be reached at full heating power. However, in most situations this cannot be measured experimentally because many actuators are dimensioned so that the only require approximately one third of the heating power for constant operation at the operating point. The power reserve is only intended to cover operating point changes and heating up phases.
- **Simulation of pressure control systems:** If you define the valve position so that it is closed at 0% and open at 100%, the process Gain of a container pressure control system is normally a negative value because the pressure reduces (>0) when the outlet valve of the container is opened. In contrast to this, the gain of an overpressure value is positive. `PV0 > 0` is the pressure when the value is completely closed. The situation is, of course, the opposite when pressures below ambient pressure are involved, for example, in vacuum systems. Note that most valves do not return a reproducible characteristic in the region of their closed position (actuation ratio 1:20 or 1:50). The time constants for pressure control of liquids are typically fast, whereas with pressure control in gas tanks, particularly in large tanks, they are slower. The magnitude of the process gain depends largely on the physical units of pressure, for example, Bar or Pa. Pressure sensors typically develop higher measurement noise than temperature sensors.
- **Simulation of flow control systems:** If you define the valve position so that it is closed at 0% and open at 100%, the process `Gain` is usually positive, since the flow rate increases when the valve opens. `PV0 = 0` if the flow stops completely when the valve is closed, in other words, the valve closes tight. The time constants are significantly faster than in temperature controls and are usually all of the same order. The magnitude of the process gain depends largely on the physical units of flow, for example, m3/s or l/min. The measurement noise affecting flow sensors is normally higher than with temperature sensors.

To implement a dead time for process simulation, you can insert a DeadTime block before the ProcSimC input and call it in a different cyclic interrupt OB (OB3x).

### Model variants

Two different model variants are supplied:

1. Continuous process simulation ProcSimC in which the MV input is an analog value, for example, heating power or valve position.
2. Process simulation ProcSimS for step controllers, with control of the actuator by two binary inputs "up"/"down" or "open"/"close". Internally, the actuator is modeled as an integrator, whereby:
   - MotorHiLim = 100%,
   - MotorLoLim = 0%
   - TI = MotorTime.
   The integrator input is derived from the binary inputs according to the following formula:

$$\overline{Integ.Input} = \begin{Bmatrix} 100 & if\ Up = True \\ -100 & if\ Down = True \\ 0 & otherwise \end{Bmatrix}$$

### See also

NoiseGen I/Os (Page 2461)

## 22.44.3 Cascade control of temperature by using the heat flow (CascadeSim)

### Cascade control of a temperature by using the heat flow

This template contains simulation models for a flow and temperature controlled system and the parameters of the noise generator block (see the I/O table). You can use this model to test the mode transitions described in the Cascade control (Page 2840) section. You can also try out the properties of different parameter sets for the primary and secondary controllers. The following features are typical for this type of application:

- The controlled temperature system is slower than the controlled flow system.
- There are two time constants that are far apart.
- There is an offset corresponding to ambient temperature.
- It has less noise than the controlled flow system.

**Process parameters of the example project for cascade control**

| ProcSimC | Gain | TimeLag1 | TimeLag2 | PV0 | NoiceVariance |
|---|---|---|---|---|---|
| Flow control loop | 8 | 1 | 1 | 0 | 0.22 |
| Temperature control loop | 0.3 | 8 | 1 | 20 | 0.1 |

**Parameters for PID controllers --> PI cascade with fast control response**

The parameters listed in the table below apply to a fast control response with low control error but with strong actuator intervention.

| PID | Gain | TI | TD |
|---|---|---|---|
| TIC101 | 10 | 8.8 | 2.6 |
| FIC101 | 0.1 | 1.8 | 0 |

**Controller parameters for PI --> P cascade with soft controller intervention**

The advantage of the parameters listed in the table below is that the controller "goes easy" on final control element (for example a valve).

| PID | Gain | TI | TD |
|---|---|---|---|
| TIC101 | 10 | 6.8 | 0 |
| FIC101 | 0.1 | 0 | 0 |

It is generally advisable to make the secondary controller "simpler" than the primary controller, in other words to reduce the number of different dynamic channels so that it is genuinely secondary to the primary controller.

The steady state control error in the secondary loop is not normally relevant for the application. On the other hand, the reaction time of the secondary loop is important because the time constants of the secondary closed control loop are part of the controlled system for the primary controller. If you do without I action in the secondary controller for these reasons, it is not advisable limit the setpoint ranges of the secondary controller precisely to the achievable physical range of the process value in the secondary loop, as you would not be able to use the full actuating range of the secondary controller due to the steady state deviation. You should instead set more generous setpoint limits for the secondary controller and manipulated variable limits for the primary controller. The anti-Windup measures of the primary controller are oriented on the interconnection of `IntHoldNeg` and `IntHoldPos`. If the secondary controller does not have any I action, if will not be capable of a bumpless manual-automatic changeover. You should therefore set an `MV_Offset` that approximates the typical `MV` value for the operating point of the process.

A cascade temperature control system with a secondary controller for heating and/or cooling medium flow is commonly used for

- Heat exchangers
- Reactors without a cooling jacket

## 22.44.4 Control loop monitoring for simulation with colored noise (ConPerMonSim)

### Control loop monitoring with simulation of colored noise

The interconnection of the ConPerMon block with a PID controller can be found in the process tag type PID_Control (see PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 2822)). The example project supports you and helps to familiarize you with the concept and the potential of closed control loop monitoring. To do this, the template includes a process simulation with disturbance model. The colored noise is generated with the aid of a shape filter from a white noise signal. This produces a spectrum of disturbance signals that also contains energy components in the lower frequency ranges of the bandwidth of the closed control loop. Part of the disturbances can therefore be compensated by the PID controller while the high-frequency measurement noise cannot be corrected by any controller.

### Application

After commissioning the controller and ConPerMon block, you should be able to watch the effects of the following actions that demonstrate the potential of control loop monitoring:

- Switch the controller to manual mode:
  The variance of the controlled variable will rise but the `CPI` becomes invalid because no statements can be made about the control fit unless the control loop is closed.
- Change the parameters of the process simulation, for example, change `TimeLag2` from 2s to 8s:
  This deterioration of the dynamic characteristics of the process (for example due to wear and tear) brings about a deterioration of the control quality that becomes visible in the `CPI` value long before it can be seen with the naked eye in the standard `PV` trends. If the control quality drops below a defined level, a `CPI` warning or even an alarm is generated.
- Request a setpoint step change from the controller:
  The `CPI` will become temporarily invalid because all stochastic characteristics of the control quality, such as the variance, are based on the assumption of a steady state with a constant mean value. Select the "Setpoint" view from the drop-down list box in the ConPerMon faceplate to be able to watch the deterministic characteristics such as overshoot and settling ratio. Once a steady state is achieved again at the new setpoint and the entire time window is filled with data from the steady state, the monitoring of the stochastic characteristics is reactivated automatically.

You can find detailed information on the ConPerMon block and notes on interpreting its displays in the online help on the block (Page 841).

## 22.44.5 Feedforward control to compensate a measurable disturbance variable (DisturbCompSim)

### Feedforward control to compensate a measurable disturbance variable

The example is based on the process tag type PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 2827) and uses the following parameter sets:

Main controlled system:

$$g(s) = \frac{2}{2s+1} e^{-1.2s}$$

Disturbance transfer function:

$$g_z(s) = \frac{1}{3s+1} e^{-1.6s}$$

- PID: Gain = 0.197
- TI= 1.9
- TD= 0

Feedforward control:

$$c(s) = -\frac{g_z(s)}{g(s)} = -\frac{1}{2} \cdot \frac{2s+1}{3s+1} e^{-0.4s}$$

The same process simulation is set up twice, one instance with disturbance feedforward and the other without (all other process and controller parameters identical). The advantages of the feedforward control can be tested in a direct comparison ("benchmark simulation", "parallel slalom ").

## 22.44.6 Operating point-oriented adaptation of parameters (gain scheduling) for non-linear processes (GainSchedSim)

### Operating point-oriented adaptation of parameters (gain scheduling) for non-linear processes

The example is based on the process tag type PID - control with operating-point-oriented parameter control (GainScheduling) (Page 2825).

In the simulation template, the settings of the two most important process parameters are changed based on polylines depending on the operating point. The process and controller parameters for the example are shown in the following table.

| Operating point | X=PV | ProcSim.Gain | ProcSim.TmLag1 | ProcSim.TmLag2 | Gain | TI | TD |
|---|---|---|---|---|---|---|---|
| 1 | 20 | 4 | 5 | 10 | 0.6 | 14.7 | 3.7 |
| 2 | 100 | 3 | 3 | 10 | 1 | 8.8 | 2.2 |
| 3 | 200 | 2 | 1 | 10 | 10 | 4.1 | 1.1 |

The same process simulation is set up twice, one instance with gain scheduling and the other without (all other process and controller parameters are identical). The advantages of the gain scheduling can be tested in a direct comparison ("benchmark simulation", "parallel slalom").

## 22.44.7 Override control on a pipeline (OverrideSim)

### Override control on a pipeline

The example is based on the process tag type Override control (Page 2846) and uses the following parameter sets:

**Primary process (flow control):**

$$g(s) = \frac{3}{(2s+1)^2}$$

Flow increases when the valve is opened and it disappears when the valve is closed.

**PI flow controller**: Gain= 0.33 , TI= 2.7

**Secondary process (pressure control):**

$$g_p(s) = \frac{-0.8}{(7s+1)(1s+1)}$$

The pressure rises when the valve is opened and is 80 bar when it is fully open.

**PI pressure controller:** Gain= 2.8 , TI= 4

Switching limits 15 bar < pressure < 70 bar.

## 22.44.8 Smith predictor for a dead time system (SmithPredictorSim)

### Smith predicator for a dead time system

The example is based on the process tag type PID controller with Smith predictor (SmithPredictorControl) (Page 2830).

In the example, the same process simulation is set up twice, one instance with Smith predictor and the other without (all other process parameters are identical). The advantages of the Smith predictor can be tested in a direct comparison ("benchmark simulation", "parallel slalom").

## 22.44.9 Filtering of noisy measured values in a control loop (SigSmoothSim)

### Filtering of noisy measured values in a control loop

The example illustrates the use of the Smooth block in a closed control loop. The block can be connected to any signal source without specialist knowledge so there is no need for a special process tag type. The simulation template is useful in testing the effects of a low-pass filter on a closed control loop by simulation. Increasing the filter time constant improves the smoothing effect but also causes a phase lag in the control loop that can have detrimental effects on the control quality and even the stability.

## Parameters used

The following parameters are used in the simulation example:

**Process transfer function:**

$$g(s) = \frac{3}{(15s+1)(2s+1)}$$

with white noise on the output signal.

**PI controller:**

- Gain = 0,5
- TI = 7 s
- Sample time = 0.1s

**Butterworth filter:**

- TimeConstant = 3 s.

At 0.3 seconds, hardly any smoothing effect can be recognized, at 15 seconds significant deterioration of the control quality is already noticeable.

Processes with signals strongly affected by noise are a typical area of application (for example pressure sensors) and sensitive actuators (for example valves).

You can find detailed information on the Smooth block in the online help on the block (Page 2297).

## 22.44.10 Predictive control of a 2x2 multi-variable controlled system (ModPreConSim)

### Predictive control of a 2x2 multi-variable controlled system

The example is based on the process tag type Model-based predictive control (ModPreCon) (Page 2849).

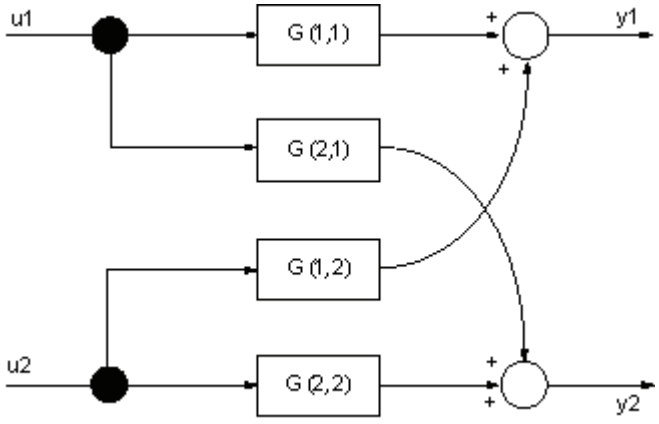



Figure 22-7 MIMO 2x2 process with a p-canonical structure

The example shows the application of the ModPreCon block for simulating a 2x2 multi-variable process consisting of the following four transfer functions:

$$\underline{G}(s) = \begin{bmatrix} G(1,1) & G(1,2) & \cdots & G(1,n_u) \\ G(2,1) & G(2,2) & \cdots & G(2,n_u) \\ \vdots & \vdots & \ddots & \vdots \\ G(n_y,1) & G(n_y,2) & \cdots & G(n_y,n_u) \end{bmatrix} = \begin{bmatrix} \dfrac{3}{(30s+1)(4s+1)} & \dfrac{1.2}{(34s+1)(14s+1)(6s+1)} \\ \dfrac{1.3}{(28s+1)(12s+1)(6s+1)} & \dfrac{4}{(26s+1)(6s+1)} \end{bmatrix}$$

where $n_y$ = 2 = number of controlled variables, $n_u$ = 2 = number of manipulated variables, and $G(i_y, i_u)$ the transfer function from input $i_u$ to output $i_y$. This simplest of multi-variable control systems helps familiarize newcomers with the concept and application of model-based multi-variable controllers.

## 22.44.11 Predictive control of a non-linear process (ModPreConNonLinSim)

### Predictive control of a non-linear process

The example is based on the approach used by multi-model controlling as described in the section ModPreCon functions (Page 976), Controlling linear and non-linear processes.

A multi-variable process is observed with two input variables and two output variables. The non-linear reaction of four partial transfer functions Proc511, Proc512, Proc521 and Proc522 depends on a measurable process value, in this case the controlled variable PV511.

The assumption that all non-linearities of the multi-variable process depend on the current operating point, which is defined by a single measurable variable, restricts the range of application but is reasonable in many practical applications. The approach of the presented multi-model controlling only makes sense with this assumption.

In the example it is assumed that the operating point is defined by a temperature, and the process reaction is different at high temperatures (200° C) than it is at low temperatures (20° C).

Some parameters of the third order partial transfer function

$$Proc(i, j) = \frac{CV(i)}{MV(j)} = \frac{Gain}{(TmLag1 \cdot s + 1) \cdot (TmLag2 \cdot s + 1)(TmLag3 \cdot s + 1)}$$

are set using polylines continually depending on the operating point.

The extreme values of the operating-point-oriented parameters for the example are shown in the following table:

| Operating point | PV511 | Proc511.Gain | Proc511.TmLag2 | Proc521.Gain | Proc512.TmLag2 | Proc522.Gain |
|---|---|---|---|---|---|---|
| 1 | 20 | 4 | 12 | 0,9 | 19 | 3 |
| 2 | 200 | 2 | 2 | 1,7 | 9 | 5 |

The changes of the process parameters are so substantial that a single linear controller cannot achieve enough control performance over the entire operating range. The changes, however, are continual and reproducible, which is an important requirement for the multi-model approach.

All other process parameters are constant:

| | Gain | TmLag1 | TmLag2 | TmLag3 |
|---|---|---|---|---|
| **Proc511** | variable | 30 | variable | 0 |
| **Proc512** | 1,2 | 34 | variable | 6 |
| **Proc521** | variable | 28 | 12 | 6 |
| **Proc522** | variable | 26 | 6 | 0 |

The primary controller TIC511522Low was designed using the MPC Configurator for the low operating point at 20° C, the secondary controller TIC511522High is made for the high operating point at 200° C.

The matching functions for relevance of the two controllers are polylines with four interpolation points at 0, 30, 190 and 300° C. In the range between 0 and 30° C, only the controller designed for 20° C is active; only the controller designed for 200° C is active between 190 and 300° C. The manipulated variables of both controllers overlap between 30 and 190°.

# Definitions

# 23

## 23.1 Batch process

**Batch process**

A batch process is a process control process, which is executed according to a recipe control batch by batch, i.e. intermittently, in a continually repeating sequence, for example, dosing raw material, tempering, performing chemical reactions, cooling, discharging reactors.

## 23.2 Approximation

### Approximation

An approximation method in the mathematical sense.

# 23.3 Prediction horizon

**Prediction horizon**

For predictive controller: Time period running from the present to the future with a defined length. A process reaction is predicted within the prediction horizon.

## 23.4 Trajectory

### Trajectory

In physics: refers to a flight path or track.

In control engineering: course of a variable over time, described by a sequence of values in a specified time scale.

## 23.5 Maverick

### Maverick

A maverick in a continuous physical measurement is a numerical value that changes from one sampling point to another more than would be physically plausible. In other words, the difference between two neighboring values is greater than a specified tolerance range.

# 23.6 Ergodic process

## Ergodic process

An ergodic process in mathematical statistics is a stationary process, in which the expected value can be estimated by generating the mean value over a time period of infinite length.

# 23.7 Conti process

## Conti process

A Conti process is a process control process, whereby raw material is fed in a continual flow, and the products are continually output.

# 23.8 Multivariable controller

## Multivariable controller

With a multivariable controller, one manipulated variable can influence several controlled variables and one controlled variable can be influenced by several manipulated variables, as is shown in the following diagram.

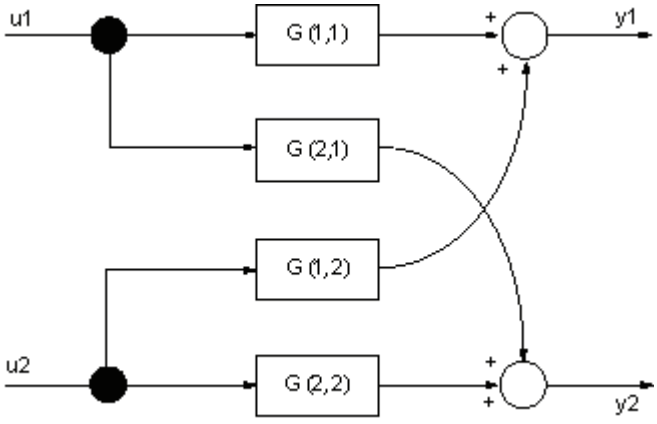



Figure 23-1 Example of a multivariable controller

When a multivariable section is automated using several individual PID controllers, the individual controllers do not take account of the interactions or links in the process. The stronger the links between the sub-sections, the harder it is to set individual controllers and the worse the control performance.

In such cases a multivariable controller offers higher control performance and simpler controller settings.

# 23.9 non-phase minimum

### Non-phase minimum behavior mean

A phase minimum system is described by a linear, time-invariant transfer function, the frequency of which displays the smallest possible clockwise phase rotation for the given number of poles and zeros if the frequency interval is run through from negative to positive in full an infinite number of times. This means that both the transfer function and its inverse are stable.

Non-phase minimum behavior means for example that the process initially deflects downwards when the manipulated variable jumps positively before moving in a positive direction. Dead time systems are also non-phase minimal.

# Index


## 0

0-1 edge transition, 85, 188

## 2

2-way valve circuit, 2866

## A

AbsR
  Area of application, 2207
  Block diagram, 2209
  Configuration, 2207
  Error handling, 2208
  Forming the signal status for blocks, 2208
  Functions, 2208
  How it works, 2207
  I/Os, 2208
  Messaging, 2208
  Object name, 2207
  Operating modes, 2208
  Startup characteristics, 2207
  Status word allocation, 2207
Activate and deactivate maverick detection
  Smooth, 2299
Activating error state for external process control error
CSF, 162
Activation and deactivation of messages
  Event, 2020
  Event16Ts, 2062
  EventNck, 2033
  EventTs, 2046
Actuating signal, 846
Actuator, 1094, 1141, 2831, 2840, 2878
Actuator active information
  FmCont, 884
  FmTemp, 924
  PIDConL, 1025
  PIDConR, 1105
  PIDConS, 1074
  PIDStepL, 1150
  VlvAnL, 1829
Actuator block, 2832
Actuators, 2833, 2872
Adapting the color representation in the configured message class
  MonDiL, 775
  MonDiS, 800
Add04
  Area of application, 2210
  Block diagram, 2214
  Configuration, 2210
  Error handling, 2212
  Forming the signal status for blocks, 2211
  Functions, 2211
  How it works, 2210
  I/Os, 2213
  Messaging, 2212
  Object name, 2210
  Operating modes, 2211
  Startup characteristics, 2210
  Status word allocation, 2210
Add08
  Area of application, 2215
  Block diagram, 2219
  Configuration, 2215
  Error handling, 2217
  Forming the signal status for blocks, 2216
  Functions, 2216
  How it works, 2215
  I/Os, 2218
  Messaging, 2217
  Object name, 2215
  Operating modes, 2216
  Startup characteristics, 2215
  Status word allocation, 2215
Adding additional values, 349
AddInt64
  Area of application, 2787
  Object name, 2787
Additional analog value
  Limit monitoring, 98
AddR64
  Area of application, 2788
  Object name, 2788
Advanced process control structures, 2870
Alarm delay
  Blocks with one time value per limit pair, 226
  Blocks with two time values per limit pair, 227
Alarm delays with a time value for all limits
  ConPerMon, 856
Alarm delays with one time value per limit pair
  AV, 707

MonAnS, 753
MotSpdCL, 1496
VlvAnL, 1826
Alarm delays with two time values per limit pair
DoseL, 1322
Alarm thresholds, 226, 227, 229
Alternatives for determining the benchmark
ConPerMon, 851
Analog driver blocks, 2870
AND operation, 85
And04
Area of application, 2399
Block diagram, 2402
Configuration, 2399
Error handling, 2401
Functions, 2400
How it works, 2399
I/Os, 2401
Messaging, 2401
Object name, 2399
Operating modes, 2400
Startup characteristics, 2399
Status word allocation, 2399
And08
Area of application, 2404
Block diagram, 2408
Configuration, 2404
Error handling, 2405
Functions, 2405
How it works, 2404
I/Os, 2406
Messaging, 2406
Object name, 2404
Operating modes, 2405
Startup characteristics, 2404
Status word allocation, 2404
Anti-windup
FmCont, 887
FmTemp, 928
ModPreCon, 979
MPC10x10, 1246
PIDConL, 1028
PIDConR, 1112
PIDConS, 1076
PIDStepL, 1153
AOTC, 349
Adding digital values, 344
Adding values, 344
AOTC window, 345
Deleting a trend, 354
Enabling/disabling a trend, 349
Enabling/disabling a value axis, 350
Opening, 343
Opening a related faceplate, 349
Opening a separate message window, 353
Saving and reopening a trend group, 351
Area of application
AbsR, 2207
Add04, 2210
Add08, 2215
AddInt64, 2787
AddR64, 2788
And04, 2399
And08, 2404
AssetM, 2778
AV, 705
Average, 2220
CompAn02, 2319
ConPerMon, 841
CountOh, 2098
CountScL, 2075
DeadTime, 2226
Derivative, 2233
DiToInt64, 2789
DoseL, 1305
Event, 2017
Event16Ts, 2057
EventNck, 2030
EventTs, 2042
FbAnIn, 2466
FbAnOu, 2476
FbAnTot, 2487
FbDiIn, 2495
FbDiOu, 2505
FbEnMe, 2528
FbSwtMMS, 2541
FirstIn, 2012
FlipFlop, 2409
FlowCorr, 2245
FmCont, 876
FmTemp, 916
GainSched, 959
Int64ToDi, 2790
Integral, 2253
Intlk02, 1949
Intlk04, 1962
Intlk08, 1976
Intlk16, 1992
KalFilt, 1276
Lag, 2261
Limit, 2324
MeanTime, 2268
MemR256, 2791
ModPreCon, 970

MonAnL, 715
MonDi08, 816
MonDiL, 770
MonDiS, 796
MotL, 1376
MotRevL, 1440
MotS, 1415
MotSpdCL, 1488
MotSpdL, 1556
MPC10x10, 1234
MSTIn, 2738
MSTOu, 2742
Mul04, 2275
Mul08, 2281
MuxAn03, 2330
MuxAn08, 2338
MuxMST, 2765
MuxST, 2770
NegInt64, 2792
NegR64, 2793
NoiseGen, 2461
Not01, 2427
OpAnL, 389
OpAnS, 412
OpDi01, 426
OpDi03, 441
OpStations, 457
OpTrig, 470
Or04, 2417
Or08, 2422
Pcs7AITC, 2551
Pcs7AnIn, 2564
Pcs7AnOu, 2577
Pcs7Cnt2, 2636
Pcs7Cnt3, 2646
Pcs7DiCnt, 2588
Pcs7DiIn, 2596
Pcs7DiIT, 2605
Pcs7DiOu, 2616
Pcs7HaAI, 2665
Pcs7HaAO, 2672
PIDCoefR, 2794
PIDConL, 1017
PIDConR, 1094
PIDConS, 1069
PIDKernR, 2800
PIDStepL, 1141
Polygon, 2287
PowerXY, 2313
Psc7Cnt1, 2625
R64ToReal, 2795
RateLim, 2344
Ratio, 1196
Rcv_AnaVal, 2812
Rcv_DigVal, 2805
RealToR64, 2796
RedAn02, 2352
RedDi02, 2431
SelA02In, 2357
SelA16In, 2362
SelAn16L, 2380
SelD02In, 2436
SelST16, 2797
ShLeInt64, 2798
ShrdResL, 1603
ShrdResS, 1626
ShRiInt64, 2799
Smooth, 2297
Snd_AnaVal, 2809
Snd_DigVal, 2801
SplRange, 1216
SqrRoot, 2303
STIn, 2730
STOu, 2734
StrctCom, 2441
StrctDeC, 2446
StruAnIn, 2706
StruAnOu, 2710
StruDiIn, 2714
StruDiOu, 2718
StruScIn, 2722
StruScOu, 2726
StruToBy, 2701
Sub02, 2308
TimerP, 2173
TimeTrig, 2181
TotalL, 2123
Trigger, 2451
Vlv2WayL, 1647
VlvAnL, 1808
VlvL, 1691
VlvMotL, 1754
VlvS, 1728
XOr04, 2456
AssetM, 2778
Area of application, 2778
Configurable reactions using the Feature parameter, 2781
Configuration, 2778
Description, 2778
Error handling, 2782
Functions, 2781
How it works, 2778
Signal status, 2781

Startup characteristics, 2778
Status word allocation, 2778
Associated values
AV, 711
ConPerMon, 859
CountOh, 2109
CountScL, 2085
DoseL, 1331
Event, 2024
EventNck, 2036
EventTs, 2050
FmCont, 896
FmTemp, 937
MonAnL, 728
MonAnS, 757
MonDi08, 824
MonDiL, 781
MonDiS, 805
MotL, 1393
MotRevL, 1462
MotS, 1426
MotSpdCL, 1518
MotSpdL, 1578
OpAnL, 396
PIDConL, 1038
PIDConR, 1120
PIDConS, 1082
PIDStepL, 1162
TotalL, 2139
Vlv2WayL, 1664
VlvAnL, 1834
VlvL, 1706
VlvMotL, 1778
VlvPosL, 1911, 1912
VlvS, 1739
Associated values (MsgEvId1)
Event16Ts, 2067, 2068
Associated values (MsgEvId2)
Event16Ts, 2068
Automatic closed-loop mode, 221
Automatic mode
Controller blocks, 79
Dosers, 82
Motors, 82
Valves, 82
Auxiliary values
Display, 239
AV
Alarm delays with one time value per limit pair, 707
Area of application, 705
Associated values, 711
Block diagram, 714
Configurable reactions using the Feature parameter, 709
Configuration, 705
Error handling, 709
Forming the signal status for blocks, 708
Functions, 707
Generating instance-specific messages, 708
How it works, 705
I/Os, 711
Limit monitoring of an additional analog value, 708
Limit monitoring with hysteresis, 708
Messaging, 710
Mode switchover error, 710
Operating modes, 707
Overview of error numbers, 709
Process messages, 710
Release for maintenance, 708
Selecting a unit of measure, 708
Simulating signals, 708
Startup characteristics, 705
Status word allocation, 706
Average
Area of application, 2220
Block diagram, 2225
Configurable reactions using the Feature parameter, 2222
Configuration, 2221
Error handling, 2222
Forming the signal status for blocks, 2222
Functions, 2222
How it works, 2220
I/Os, 2224
Messaging, 2223
Object name, 2220
Operating modes, 2221
Overview of error numbers, 2223
Startup characteristics, 2221
Status word allocation, 2221


## B


Batch columns, 2826
Batch execution, 961
Batch process, 848, 2881
Batch reactors, 2826
Batch view, 336
Benchmark, 847
Benchmark simulation, 2876, 2877
Block diagram
AbsR, 2209

Add04, 2214
Add08, 2219
And04, 2402
And08, 2408
AV, 714
Average, 2225
CompAn02, 2323
ConPerMon, 866
CountOh, 2114
CountScL, 2090
DeadTime, 2231
Derivative, 2239
Div02, 2244
DoseL, 1348
Event, 2029
EventNck, 2041
EventTs, 2056, 2073
FbAnIn, 2475
FbAnOu, 2486
FbAnTot, 2494
FbDiIn, 2503
FbDiOu, 2515
FbEnME, 2539
FbSwtMMS, 2550
FirstIn, 2016
FlipFlop, 2416
FlowCorr, 2252
FmCont, 913
FmTemp, 956
GainSched, 966
Integral, 2259
Intlk02, 1961
Intlk04, 1974
Intlk08, 1991
Intlk16, 2010
KalFilt, 1292
Lag, 2267
Limit, 2329
MeanTime, 2274
ModPreCon, 1001
MonAnL, 738
MonAnS, 762
MonDi08, 829
MonDiL, 786
MonDiS, 808
MotL, 1404
MotRevL, 1474
MotS, 1432
MotSpdCL, 1535
MotSpdL, 1590
MPC10x10, 1260
MSTIn, 2741
MSTOu, 2745
Mul04, 2279
Mul08, 2285
MuxAn03, 2337
MuxAn08, 2343
MuxMST, 2768
MuxST, 2774
Not01, 2430
OpAnL, 403
OpAnS, 420
OpDi01, 435
OpDi03, 451
OpStations, 464
OpTrig, 477
Or04, 2421
Or08, 2426
Pcs7AnIn, 2576
Pcs7AnOu, 2587
Pcs7Cnt3, 2654
Pcs7DiIn, 2604
Pcs7DiIT, 2614
Pcs7DiOu, 2624
Pcs7HaAI, 2671
Pcs7HaAO, 2678
PcsCnt2, 2645
PIDConL, 1058
PIDConR, 1140
PIDConS, 1091
PIDStepL, 1179
Polygon, 2296
PowerXY, 2317
Psc7Cnt1, 2635
RateLim, 2351
Ratio, 1207
RedAn02, 2356
RedDi02, 2435
SelA02In, 2361
SelA16In, 2372
SelAn16L, 2392
SelD02In, 2440
ShrdResL, 1619
ShrdResS, 1641
Smooth, 2302
SplRange, 1223
STIn, 2733
STOu, 2737
StrctCom, 2444
StrctDeC, 2449
StrgToBy, 2704
StruAnIn, 2709
StruAnOu, 2713
StruDiIn, 2717

StruDiOu, 2721
StruScIn, 2725
StruScOu, 2729
Sub02, 2307, 2312
TimerP, 2180
TimeTrig, 2197
TotalL, 2146
Trigger, 2455
Vlv2WayL, 1676
VlvAnL, 1850
VlvL, 1715
VlvMotL, 1791
VlvPosL, 1928
VlvS, 1745
XOr04, 2460
Block diagram of CntOhSc, 2165
Block diagram of Rcv_DigVal, 2808
Block diagram of RealToDw, 2749
Block diagram of Snd_DigVal, 2804, 2811, 2815
Block diagram of STRep, 2777
Block icon
Configuring, 267
ConPerMon, 874
CountOh, 2121
DoseL, 1366
Interlock block, 272
KalFilt, 1302
ModPreCon, 1014
MonDiL, 793, 813
MotRevL, 1485
MotSpdCL, 1552
MotSpdL, 1600
MPC10x10, 1274
OpAnL, 410
OpAnS, 425
OpDi01, 439
OpDi03, 455
Operating, 269
OpStations, 469
OpTrig, 480
PIDConL, 269
PIDConS, 269
Ratio, 1213
SelA16In, 2377
SelAn16L, 2397
ShrdResL, 1625
ShrdResS, 1645
Static picture component, 276
TimeTrig, 2205
TotalL, 2154
Vlv2WayL, 1688
VlvAnL, 1872
VlvL, 1725
VlvMotL, 1805
VlvPosL, 1947
VlvS, 1751
Block icon structure, 260
Block icons for CntOhSc, 2170
Block icons for SFC, 274
Block symbol
MonAnL, 748
MonAnS, 768
MonDi08, 834
MotL, 1412
MotS, 1438
Block-external simulation, 64
Block-internal simulation, 64
Blocks
Operator control permissions, 285
Bumpless, 81, 189
Bumpless switchover, 84, 130, 183, 184, 222, 316, 319
Controller blocks, 79
Dosers, 82
Manipulated variable, 139
Motors, 82
Setpoint, 137
Valves, 82
Bumpless switchover between internal and external setpoint with PID initialization
PIDConR, 1107
Bumpless switchover from external to internal ratio
Ratio, 1199
Bumpless switchover from external to internal setpoint
DoseL, 1319
PIDConR, 1107
Button labels
FmCont, 892
MotL, 1390, 1420
MotRevL, 1457
MotSpdCL, 1510
MotSpdL, 1574
PIDStepL, 1158
Vlv2WayL, 1660
VlvAnL, 1831
VlvL, 1703
VlvMotL, 1773
Bypass
Intlk02, 1953
Intlk04, 1966
Intlk08, 1980
Intlk16, 1998
Bypass function
MonAnL, 723

MonDiL, 777
PIDConL, 1026
PIDConR, 1107
PIDStepL, 1151
TotalL, 2134
Bypassing signals, 115

## C

Calculation of the flow rate for dosing by scale
DoseL, 1316
Cascade, 1196
Cascade and ratio controls, 154
Cascade circuit, 2832
Cascade control, 222, 851, 876, 916, 970, 1017, 1094, 1141, 2819, 2832, 2840, 2843, 2873
ConPerMon, 851
Cascading
ShrdResS, 1631
Changing labels on buttons and text
MonDi08, 821
MonDiL, 776
MonDiS, 801
OpDi01, 428
OpDi03, 444
Channel block, 70
Channel blocks, 148
Channel driver block, 117, 185
Channel error, 130
FbAnIn, 2471
FbAnOu, 2480
FbDiIn, 2499
FbDiIT, 2610
FbDiOu, 2509, 2620
Pcs7AITC, 2558, 2591
Pcs7AnIn, 2572
Pcs7AnOu, 2583
Pcs7DiIn, 2600
Channel function block, 233
Channel management
ShrdResL, 1609
ShrdResS, 1630
Channel prioritization
ShrdResL, 1611
ShrdResS, 1631
Checking the raw value
Pcs7AITC, 2554
CntOhSc
I/Os, 2161
Coded unit of measure, 240
CompAn02
Area of application, 2319
Block diagram, 2323
Configuration, 2319
Error handling, 2321
Forming the signal status for blocks, 2320
Functions, 2320
How it works, 2319
I/Os, 2322
Messaging, 2322
Object name, 2319
Operating modes, 2320
Startup characteristics, 2319
Status word allocation, 2320
Configurable functions using the Feature parameter
Event16Ts, 2064
EventTs, 2047
Configurable reactions using the Feature block
Event, 2021
Configurable reactions using the Feature I/O
DeadTime, 2229
IntIk04, 1969
Configurable reactions using the Feature parameter
AssetM, 2781
AV, 709
Average, 2222
ConPerMon, 855
CountOh, 2104
CountScL, 2081
Derivative, 2236
DoseL, 1323
EventNck, 2034
FbAnIn, 2470, 2498
FbAnOu, 2479
FbAnTot, 2489
FbDiOu, 2508
FbSwtMMS, 2542
FmCont, 889
FmTemp, 930
GainSched, 962
Integral, 2256
Intlk02, 1956
Intlk08, 1984
Intlk16, 2001
KalFilt, 1284
Lag, 2264
MeanTime, 2270
ModPreCon, 984
MonAnL, 723
MonAnS, 754
MonDi08, 820
MonDiL, 777
MonDiS, 802
MotL, 1388

MotRevL, 1455
MotS, 1422
MotSpdCL, 1508
MotSpdL, 1572
MPC10x10, 1249
OpAnL, 393
OpAnS, 415
OpDi01, 430
OpDi03, 446
OpTrig, 473
Pcs7AITC, 2556
Pcs7AnIn, 2570
Pcs7AnOu, 2582
Pcs7Cnt2, 2640
Pcs7Cnt3, 2649
Pcs7DiCnt, 2590
Pcs7DiIn, 2599
Pcs7DiIT, 2609
Pcs7DiOu, 2619
PcsCnt1, 2628
PIDConL, 1030
PIDConR, 1113
PIDConS, 1077
PIDStepL, 1155
RateLim, 2347
Ratio, 1201
SelA16In, 2366
SelAn16L, 2384
ShrdResL, 1607
ShrdResS, 1629
TotalL, 2135
VlvAnL, 1825
VlvL, 1701
VlvMotL, 1772
VlvPosL, 1905
VlvS, 1735
Configurable reactions using the Feature2 parameter, 1423
DoseL, 1324
FmCont, 889
FmTemp, 931
MotRevL, 1456
MotS, 1423
MotSpdCL, 1509
MotSpdL, 1573
PIDConL, 1031
PIDStepL, 1156
VlvS, 1736
Configurable reactions using the Features parameter
Vlv2WayL, 1659
Configuration
AbsR, 2207
Add04, 2210
Add08, 2215
And04, 2399
And08, 2404
AssetM, 2778
AV, 705
Average, 2221
CompAn02, 2319
ConPerMon, 842
CountOh, 2100
CountScL, 2077
DeadTime, 2227
Derivative, 2235
Div02, 2240
DoseL, 1305
Event, 2017
Event16Ts, 2058
EventNck, 2030
EventTs, 2042
FbAnIn, 2466
FbAnOu, 2476
FbAnTot, 2487
FbDiIn, 2495
FbDiOu, 2505
FbEnMe, 2528
FbSwtMMS, 2541
FirstIn, 2012
FlipFlop, 2411
FlowCorr, 2246
FmCont, 876
FmTemp, 916
GainSched, 960
Integral, 2254
Intlk02, 1949
Intlk04, 1962
Intlk08, 1976
Intlk16, 1992
KalFilt, 1279
Lag, 2262
Limit, 2325
MeanTime, 2269
ModPreCon, 972
MonAnL, 715
MonAnS, 750
MonDi08, 816
MonDiL, 771
MonDiS, 797
MotL, 1376
MotRevL, 1440
MotS, 1415
MotSpdCL, 1488
MotSpdL, 1556

MPC10x10, 1237
MSTIn, 2738
MSTOu, 2742
Mul04, 2275
Mul08, 2281
MuxAn03, 2330
MuxAn08, 2338
MuxMST, 2765
MuxST, 2770
Not01, 2427
OpAnL, 389
OpAnS, 412
OpDi01, 426
OpDi03, 441
OpStations, 457
OpTrig, 470
Or04, 2417
Or08, 2422
Pcs7AITC, 2551
Pcs7AnIn, 2564
Pcs7AnOu, 2577
Pcs7Cnt2, 2638
Pcs7Cnt3, 2647
Pcs7DiCnt, 2588
Pcs7DiIn, 2596
Pcs7DiIT, 2605
Pcs7DiOu, 2616
Pcs7HaAI, 2665
Pcs7HaAO, 2672
PIDConL, 1017
PIDConR, 1095
PIDConS, 1069
PIDStepL, 1142
Polygon, 2289
PowerXY, 2314
Psc7Cnt1, 2626
RateLim, 2344
Ratio, 1197
Rcv_AnaVal, 2813
Rcv_DigVal, 2806
RedAn02, 2352
RedDi02, 2431
SelA02In, 2357
SelA16In, 2362
SelAn16L, 2380
SelD02In, 2436
ShrdResL, 1604
ShrdResS, 1627
Smooth, 2297
Snd_DigVal, 2802, 2810
SplRange, 1217
SqrRoot, 2303
STIn, 2730
STOu, 2734
StrctCom, 2441
StrctDeC, 2446
StruAnIn, 2706
StruAnOu, 2710
StruDiIn, 2714
StruDiOu, 2718
StruScIn, 2722
StruScOu, 2726
StruToBy, 2701
Sub02, 2308
TimerP, 2173
TimeTrig, 2181
TotalL, 2130
Trigger, 2451
Vlv2WayL, 1647
VlvAnL, 1808
VlvL, 1691
VlvMotL, 1754
VlvPosL, 1876
VlvS, 1728
XOr04, 2456
Configured runtime monitoring, 88
ConPerMon
Alarm delays with a time value, 856
Alternatives for determining the benchmark, 851
Area of application, 841
Associated values, 859
Block diagram, 866
Block icon, 874
Cascade control, 851
Configurable reactions using the Feature parameter, 855
Configuration, 842
Error handling, 857
Feedforward control, 852
Forming the signal status for blocks, 854
Functions, 845
Generating instance-specific messages, 856
How it works, 841
I/Os, 860
Instance-specific messages, 859
Limit operation and display in the faceplate, 856
Limit view, 869
Messaging, 857
Monitoring of deterministic characteristics of the control performance, 848
Monitoring of stochastic characteristics of the control performance, 846
Multivariable controller, 853
Object name, 841

Opening additional faceplates, 856
Operating modes, 845
Operator permissions, 855
Override control, 852
Overview of error numbers, 857
Parameter view, 870
PID controller with gain scheduler, 852
Preview, 871
Process messages, 858
Ratio control, 853
Selecting a unit of measure, 854
SIMATIC BATCH functionality, 856
Smith predictor, 852
Split-range control, 852
Standard view, 867
Startup characteristics, 843
Status word allocation, 844
Suppressing messages using the MsgLock parameter, 856
Considering bad quality of automatic commands or external values
DoseL, 1322
MotL, 1385
MotRevL, 1451
MotS, 1422
MotSpdCL, 1503
MotSpdL, 1567
OpAnL, 392
OpAnS, 414
OpDi01, 429
OpDi03, 445
OpTrig, 472
Vlv2WayL, 1657
VlvAnL, 1820
VlvL, 1698
VlvMotL, 1768
VlvPosL, 1899
VlvS, 1735
Conti process, 848, 960, 2887
Conti reactors, 980, 1246
Continuous controller, 876, 882, 916, 922
Control loop is open (CascaCut), 1029, 1112, 1153, 1767, 1819, 1898
Control of linear and non-linear systems
ModPreCon, 982
MPC10x10, 1248
Control of square and non-square systems
ModPreCon, 981
MPC10x10, 1247
Control outputs
DoseL, 1314
Control performance, 841, 847, 857, 868, 959, 972, 1236, 2830, 2875, 2877, 2880
Control performance index, 869
Control performance index (CPI), 847
Control quality monitoring, 2849
Control system fault
VlvPosL, 1910
Control system fault (CSF), 893, 934, 1035, 1079, 1080, 1117, 1118, 1159, 1160, 1327, 1390, 1391, 1424, 1458, 1460, 1512, 1515, 1574, 1576, 1661, 1662, 1704, 1705, 1737, 1775, 1776, 1832, 1833, 1908, 1909
MonAnL, 726, 779
MonAnS, 756
MonDiS, 803
Control zone
PIDConL, 1029
Using, 220
Control zone width, 221
Control zone with frozen I component, 171, 172
Controlled closed-loop mode, 221
Controller blocks
Automatic mode, 79
Bumpless switchover, 79
Manual mode, 79
Controller with I component, 846
Controllers
Program mode, 85
Controlling a device infeed or an inverter enable
MotSpdCL, 1510
Conversion block, 233
CountOh
Area of application, 2098
Associated values, 2109
Block diagram, 2114
Block icon, 2121
Configurable reactions using the Feature parameter, 2104
Configuration, 2100
Display and operator input area for process values and setpoints, 2105
Error handling, 2107
Forming the signal status for blocks, 2105
Functions, 2103
How it works, 2098
I/Os, 2109
Limit monitoring of the operating time, 2103
Limit view, 2118
Messaging, 2108
Object name, 2098
Opening additional faceplates, 2105
Operating modes, 2102

Operator permissions, 2106
Overview of error numbers, 2107
Parameter view, 2119
Preview, 2120
Process messages, 2108
Read back the last counted value, 2104
Release for maintenance, 2107
Reset counter to zero, 2105
Setting the count value to the default setting, 2105
SIMATIC BATCH functionality, 2107
Standard view, 2116
Startup characteristics, 2100
Status word allocation, 2100
Suppressing messages using the MsgLock parameter, 2103
Time response, 2100
CountScL
Area of application, 2075
Associated values, 2085
Block diagram, 2090
Configurable reactions using the Feature parameter, 2081
Configuration, 2077
CountScL block icon, 2096
Error handling, 2084
Forming the signal status for blocks, 2081
Functions, 2080
How it works, 2076
I/Os, 2085
Limit monitoring of the count value, 2080
Limit view, 2093
Messaging, 2084
Object name, 2075
Opening additional faceplates, 2083
Operating modes, 2079
Operator permissions, 2082
Overview of error numbers, 2084
Parameter view, 2094
Preview, 2095
Process messages, 2085
Read back the last counted value, 2080
Release for maintenance, 2083
Reset counter to zero, 2080
Selecting a unit of measure, 2081
Setting the count value to the default setting, 2081
SIMATIC BATCH functionality, 2083
Standard view, 2091
Startup characteristics, 2077
Status word allocation, 2078
Suppressing messages using the MsgLock parameter, 2081
Time response, 2077
CSF and ExtMsgx
Group display for limit monitoring, 92
Customer-specific units, 240
CV bands, 979, 1245
Cycle counter, 2220


## D


D action, 224
D component, 222, 887, 1017, 1069, 1094, 1141
Dead band
Description, 67
MonAnL, 722
MonAnS, 753
Dead band zone, 1216
DeadTime
Area of application, 2226
Block diagram, 2231
Configurable reactions using the Feature I/O, 2229
Configuration, 2227
Error handling, 2229
Forming the signal status for blocks, 2229
Functions, 2228
How it works, 2226
I/Os, 2230
Messaging, 2230
Object name, 2226
Operating modes, 2228
Overview of error numbers, 2229
Startup characteristics, 2227
Status word allocation, 2228
Deenergized state, 54
Defining valve positions for individual valves
Vlv2WayL, 1653
Delay of alarms
Event, 2020
EventNck, 2033
Delayed output of a single trigger
TimeTrig, 2184
Delaying the on and off switching functions
MonDiL, 775
Delaying the on function
MonDiS, 800
Derivative
Area of application, 2233
Block diagram, 2239
Configurable reactions using the Feature parameter, 2236

Configuration, 2235
Error handling, 2237
Forming the signal status for blocks, 2236
Functions, 2236
How it works, 2233
I/Os, 2238
Limit monitoring, 2236
Messaging, 2237
Object name, 2233
Operating modes, 2235
Overview of error numbers, 2237
Startup characteristics, 2235
Status word allocation, 2235
Description
Pcs7Cnt3, 2646
Description of, 2625, 2636
AssetM, 2778
MemR256, 2791
Pcs7Cnt2, 2636
Psc7Cnt1, 2625
StateMap, 2750
Description of CntOhSc, 2156
Description of RealToDw, 2747
Description of STRep, 2775
Detecting the creep rate, 179
Determining the dosing quantity when dosing using scales
DoseL, 1315
Determining the dosing quantity when using flow dosing
DoseL, 1314
Deterministic characteristics, 841, 847, 2875
Digital feedback from the readback value
VlvAnL, 1826
Disable feedback
VlvPosL, 1906
Disable interlocks
VlvPosL, 1897
Disabling feedback
Vlv2WayL, 1655
VlvAnL, 1830
VlvL, 1699
VlvMotL, 1769
Disabling interlocks
DoseL, 1320
MotL, 1384
MotRevL, 1449
MotS, 1421
MotSpdCL, 1502
MotSpdL, 1565
Vlv2WayL, 1656
VlvL, 1697
VlvMotL, 1766
VlvS, 1733
Disabling of limits
ModSpdCL, 1507
MonAnL, 723
VlvAnL, 1823
VlvMotL, 1771
Display and operator input area for process values and setpoints
CountOh, 2105
DoseL, 1322
Ratio, 1199
Display auxiliary values
VlvPosL, 1906
Display of selected value
SelD02In, 2437
Displaying additional information relating to the manipulated variable on the output
PIDConR, 1105
Displaying and outputting the signal status
Event16Ts, 2064
EventTs, 2047
Displaying auxiliary values
DoseL, 1326
MonAnL, 721
MonDiL, 776
MotL, 1390
MotRevL, 1457
MotSpdCL, 1510
MotSpdL, 1574
Vlv2WayL, 1660
VlvAnL, 1826
VlvL, 1703
VlvMotL, 1773
DiToInt64
Area of application, 2789
Object name, 2789
Div02
Area of application, 2240
Block diagram, 2244
Configuration, 2240
Error handling, 2242
Forming the signal status for blocks, 2241
Functions, 2241
How it works, 2240
I/Os, 2243
Messaging, 2243
Object name, 2240
Operating modes, 2241
Overview of error numbers, 2242
Startup characteristics, 2240
Status word allocation, 2240

DMC procedure (Dynamic Matrix Control), 971, 1236
DoseL
Alarm delays with two time values per limit pair, 1322
Area of application, 1305
Associated values, 1331
Block diagram, 1348
Block icon, 1366
Bumpless switchover from external to internal setpoint, 1319
Calculation of the flow rate for dosing by scale, 1316
Configurable reactions using the Feature parameter, 1323
Configurable reactions using the Feature2 parameter, 1324
Configuration, 1305
Considering bad quality of automatic commands or external values, 1322
Control outputs, 1314
Determining the dosing quantity when dosing using scales, 1315
Determining the dosing quantity when using flow dosing, 1314
Disabling interlocks, 1320
Display and operator input area for process values and setpoints, 1322
Displaying auxiliary values, 1326
Dribbling, 1316
Error handling, 1327
External/internal setpoint specification, 1318
Forcing operating modes, 1320
Forming the group status for interlocks, 1321
Forming the signal status for blocks, 1321
Functions, 1312
Generating instance-specific messages, 1323
Group error, 1321
How it works, 1305
I/Os, 1332
Instance-specific messages, 1330
Interlocks, 1320
Limit monitoring of the process value, 1319
Messaging, 1329
Mode switchover error, 1328
Object name, 1305
Opening additional faceplates, 1323
Operating modes, 1310
Operator permissions, 1324
Output signal as a pulse signal or static signal, 1314
Outputting a signal for start readiness, 1321
Overdosing/underdosing, 1317
Overview of error numbers, 1327
Post dosing, 1318
Preview, 1363
Process control fault, 1329
Process messages, 1330
Release for maintenance, 1322
Resetting the block in case of interlocks or errors, 1321
Resetting the dosing quantity, 1318
Selecting a unit of measure, 1323
Setpoint limitation, 1319
Setpoint view, 1361
SIMATIC BATCH functionality, 1326
Simulating signals, 1320
Startup characteristics, 1306
Status diagram, 1312
Status word allocation, 1306
Suppressing messages using the MsgLock parameter, 1320
Time-stamped messages, 1326
Dribbling
DoseL, 1316

## E

Emergency stop for motors, 115
Enable/disable channel
ShrdResL, 1611
ShrdResS, 1631
Ergodic process, 2886
Error handling
AbsR, 2208
Add04, 2212
Add08, 2217
And04, 2401
And08, 2405
AssetM, 2782
AV, 709
Average, 2222
CompAn02, 2321
ConPerMon, 857
CountOh, 2107
CountScL, 2084
DeadTime, 2229
Derivative, 2237
Div02, 2242
DoseL, 1327
Event, 2022
EventNck, 2035
EventTs, 2048, 2065
FbAnIn, 2471
FbAnOu, 2480

FbAnTot, 2490
FbDiIn, 2499
FbDiOu, 2509
FbEnMe, 2531
FbSwtMMS, 2543
FirstIn, 2014
FlipFlop, 2413
FlowCorr, 2249
FmCont, 892
FmTemp, 934
GainSched, 963
Integral, 2257
Intlk02, 1956
Intlk04, 1969
Intlk08, 1984
Intlk16, 2002
KalFilt, 1286
Lag, 2264
Limit, 2327
MeanTime, 2272
ModPreCon, 989
MonAnL, 726
MonAnS, 755
MonDi08, 822
MonDiL, 779
MonDiS, 803
MotL, 1390
MotRevL, 1457
MotS, 1424
MotSpdCL, 1512
MotSpdL, 1574
MPC10x10, 1253
MSTIn, 2739
MSTOu, 2744
Mul04, 2278
Mul08, 2283
MuxAn03, 2334
MuxAn08, 2341
MuxMST, 2767
MuxST, 2772
Not01, 2428
OpAnL, 394
OpAnS, 416
OpDi01, 431
OpDi03, 446
OpStations, 461
OpTrig, 473
Or04, 2419
Or08, 2424
Overview, 128
Pcs7AITC, 2558
Pcs7AnIn, 2571
Pcs7AnOu, 2582
Pcs7Cnt2, 2641
Pcs7DiCnt, 2591
Pcs7DiIn, 2600
Pcs7DiIT, 2610
Pcs7DiOu, 2620
Pcs7HaAI, 2667
Pcs7HaAO, 2674
PcsCnt3, 2649
PIDConL, 1034
PIDConR, 1117
PIDConS, 1079
PIDStepL, 1159
Polygon, 2291
PowerXY, 2315
Psc7Cnt1, 2629
RateLim, 2348
Ratio, 1201
Rcv_AnaVal, 2814
Rcv_DigVal, 2807
RedAn02, 2354
RedDi02, 2433
SelA02In, 2359
SelA16In, 2367
SelAn16L, 2385
SelD02In, 2438
ShrdResL, 1613
ShrdResS, 1632
Smooth, 2299
Snd_AnaVal, 2810
Snd_DigVal, 2803
SplRange, 1221
SqrRoot, 2305
StateMap, 2751
STIn, 2731
STOu, 2735
StrctCom, 2443
StrctDeC, 2448
StrgToBy, 2703
StruAnIn, 2707
StruAnOu, 2711
StruDiIn, 2715
StruDiOu, 2719
StruScIn, 2723
StruScOu, 2727
Sub02, 2310
TimerP, 2177
TimeTrig, 2191
TotalL, 2138
Trigger, 2453
Vlv2WayL, 1660
VlvAnL, 1832

VlvL, 1703
VlvMotL, 1774
VlvPosL, 1908
VlvS, 1736
XOr04, 2458
Error handling of RealToDw, 2748
Error handling of STRep, 2776
Error numbers
Tabular overview, 128
Error signal, 102
Error signal generation and dead band
FmCont, 885
FmTemp, 926
ModPreCon, 978
MPC10x10, 1245
PIDConL, 1026
PIDConR, 1108
PIDConS, 1075
PIDStepL, 1151
Event
Activation and deactivation of messages, 2020
Area of application, 2017
Associated values, 2024
Block diagram, 2029
Configurable reactions using the Feature block, 2021
Configuration, 2017
Delay of alarms, 2020
Error handling, 2022
Forming the signal status for blocks, 2022
Functions, 2020
How it works, 2017
I/Os, 2025
Messaging, 2023
Object name, 2017
Operating modes, 2020
Operator control permissions, 2021
Overview of error numbers, 2022
Process messages, 2023
Release for maintenance, 2021
Startup characteristics, 2017
Status word allocation, 2018
Suppressing messages using the MsgLock parameter, 2021
Event16Ts
Activation and deactivation of messages, 2062
Area of application, 2057
Associated values (MsgEvId1), 2067, 2068
Associated values (MsgEvId2), 2068
Configurable functions using the Feature parameter, 2064
Configuration, 2058
Displaying and outputting the signal status, 2064
Functions, 2062
How it works, 2057
Messaging, 2065
Object name, 2057
Operating modes, 2062
Operator permissions, 2063
Process messages (MsgEvId1), 2066
Process messages (MsgEvId2), 2067
Release for maintenance, 2063
Signal status as associated value of a message, 2063
Startup characteristics, 2058
Status word allocation, 2059
Suppressing messages using the MsgLock parameter, 2063
Time stamp as associated value of a message, 2063
EventNck
Activation and deactivation of messages, 2033
Area of application, 2030
Associated values, 2036
Block diagram, 2041
Configuration, 2030
Delay of alarms, 2033
Error handling, 2035
Forming the signal status for blocks, 2034
Functions, 2033
How it works, 2030
I/Os, 2037
Messaging, 2036
Object name, 2030
Operating modes, 2032
Operator control permissions, 2034
Overview of error numbers, 2035
Process messages, 2036
Release for maintenance, 2034
Startup characteristics, 2030
Status word allocation, 2031
Suppressing messages using the MsgLock parameter, 2033
EventTs
Activation and deactivation of messages, 2046
Area of application, 2042
Associated values, 2050
Block diagram, 2056, 2073
Configurable functions using the Feature parameter, 2047
Configuration, 2042
Displaying and outputting the signal status, 2047
Error handling, 2048, 2065
Functions, 2046

How it works, 2042
I/Os, 2052, 2071
Messaging, 2049
Object name, 2042
Operating modes, 2045
Operator permissions, 2047
Overview of error numbers, 2048, 2065
Process messages, 2049
Release for maintenance, 2046
Signal status as associated value of a message, 2046
Startup characteristics, 2043
Status word allocation, 2043
Suppressing messages using the MsgLock parameter, 2046
Time stamp as associated value of a message, 2046
External maniputaed variable, 134
External process control error, 128
External setpoint, 133
External simulation, 65
External/internal setpoint specification
DoseL, 1318
FmCont, 885
FmTemp, 925
MotSpdCL, 1499
PIDConL, 1025
PIDConR, 1106
PIDConS, 1075
PIDStepL, 1150


## F


FbAnIn
Area of application, 2466
Block diagram, 2475
Channel error, 2471
Configurable reactions using the Feature parameter, 2470, 2498
Configuration, 2466
Error handling, 2471
Flutter suppression, 2469
Functions, 2468
Higher-level error / invalid measuring range, 2471
Holding the last value if raw value is invalid, 2468
How it works, 2466
I/Os, 2472
Messaging, 2472
Object name, 2466
Obtaining the standard value, 2468
Operating modes, 2468
Output substitute value if raw value is invalid, 2469
Outputting an invalid value if analog value is invalid, 2469
Redundancy, 2467
Signal status for Fb channel blocks, 2469
Simulating signals, 2469, 2498
Startup characteristics, 2467
Status word allocation, 2467
FbAnOu
Area of application, 2476
Block diagram, 2486
Channel error, 2480
Configurable reactions using the Feature parameter, 2479
Configuration, 2476
Error handling, 2480
Flutter suppression, 2478
Functions, 2478
Higher-level error / invalid measuring range, 2480
How it works, 2476
I/Os, 2481
Keep last valid raw value, 2479
Messaging, 2480
Modes, 2478
Object name, 2476
Obtaining the standard value, 2478
Redundancy, 2477
Signal status for Fb channel blocks, 2478
Simulating signals, 2479
Startup characteristics, 2477
Status word allocation, 2477
FbAnTot
Area of application, 2487
Block diagram, 2494
Configurable reactions using the Feature parameter, 2489
Configuration, 2487
Error handling, 2490
Flutter suppression, 2489
Functions, 2488
Holding the last value if raw value is invalid, 2488
How it works, 2487
I/Os, 2491
Messaging, 2491
Object name, 2487
Obtaining the standard value, 2488
Operating modes, 2488
Output substitute value if raw value is invalid, 2488
Outputting an invalid value if analog value is invalid, 2489

Signal status for Fb channel blocks, 2489
Simulating signals, 2489
Startup characteristics, 2488
FbDiIn
Area of application, 2495
Block diagram, 2503
Channel error, 2499
Configuration, 2495
Error handling, 2499
Flutter suppression, 2497
Functions, 2497
Higher-level error / invalid measuring range, 2499
Holding the last value if raw value is invalid, 2497
How it works, 2495
I/Os, 2500
Messaging, 2500
Object name, 2495
Obtaining the standard value, 2497
Operating modes, 2496
Output of invalid value if raw value is invalid, 2497
Output substitute value if raw value is invalid, 2497
Signal status for Fb channel blocks, 2497
Startup characteristics, 2496
Status word allocation, 2496
FbDiIT
Channel error, 2610
Higher-level error / invalid measuring range, 2610
FbDiOu
Area of application, 2505
Block diagram, 2515
Channel error, 2509, 2620
Configurable reactions using the Feature parameter, 2508
Configuration, 2505
Error handling, 2509
Flutter suppression, 2507
Functions, 2507
Higher-level error / invalid measuring range, 2509
How it works, 2505
I/Os, 2510
Keep last valid raw value, 2508
Messaging, 2510
Modes, 2506
Object name, 2505
Obtaining the standard value, 2507
Signal status for Fb channel blocks, 2507
Simulating signals, 2508
Startup characteristics, 2506
Status word allocation, 2506
FbDrive, 2859
Area of application, 2516
Configuration, 2516
Error handling, 2519
Functions, 2517
How it works, 2516
I/Os, 2520
Startup characteristics, 2517
FbEnMe
Area of application, 2528
Configuration, 2528
Error handling, 2531
Functions, 2530
How it works, 2528
I/Os, 2532
Messaging, 2532
Object name, 2528
Operating modes, 2529
Startup characteristics, 2529
Status word allocation, 2529
FbEnME
Block diagram, 2539
FbSwtMMS, 2864
Area of application, 2541
Block diagram, 2550
Configurable reactions using the Feature parameter, 2542
Configuration, 2541
Error handling, 2543
Functions, 2542
How it works, 2541
I/Os, 2545
Messaging, 2544
Operating modes, 2542
Startup characteristics, 2542
Transmission of messages, 2542
Feature
Bit assignment, 139
Control zone with frozen I component, 171, 172
Description, 139
First-in detection response to deactivation, 187
Frequency converter with separate device feed, 162
Process value with separate scale range, 173
Set startup characteristics, 148
Specifying the reaction to exiting local mode, 189
Feedback monitoring, 104
MotL, 1386
MotRevL, 1451
MotS, 1422
MotSpdCL, 1504
MotSpdL, 1567

Vlv2WayL, 1654
VlvAnL, 1820
VlvL, 1699
VlvMotL, 1768
VlvPosL, 1900
VlvS, 1735
Feedbacks
Disable monitoring, 104
Monitoring, 104
Feedforward control, 2827
ConPerMon, 852
Feedforward control and limitation, 223
Feedforwarding and limiting disturbance variables
FmCont, 887
FmTemp, 928
PIDConL, 1028
PIDConR, 1112
PIDStepL, 1153
Final controlling element, 846, 884, 924, 1017, 1069, 2819, 2832, 2846, 2847, 2848, 2873
FirstIn
Area of application, 2012
Block diagram, 2016
Configuration, 2012
Error handling, 2014
Functions, 2014
How it works, 2012
I/Os, 2015
Messaging, 2015
Object name, 2012
Operating modes, 2013
Startup characteristics, 2013
Status word allocation, 2013
First-in detection
Intlk02, 1953
Intlk04, 1968
Intlk08, 1983
Intlk16, 2000
First-in detection response to deactivation, 187
Feature parameter, 187
Fixed setpoint control, 876, 916, 970, 1017, 1069, 1094, 1141
FlipFlop
Area of application, 2409
Block diagram, 2416
Configuration, 2411
Error handling, 2413
Functions, 2412
How it works, 2409
How the block works as RS-FlipFlop (Mode = 1), 2410
How the block works as SR-FlipFlop (Mode = 0), 2409
I/Os, 2414
Initialization, 2413
Messaging, 2414
Object name, 2409
Operating modes, 2412
Startup characteristics, 2411
Status word allocation, 2411
Flow alarm, 147
FlowCorr
Area of application, 2245
Block diagram, 2252
Configuration, 2246
Error handling, 2249
Forming the signal status for blocks, 2248
Functions, 2247
How it works, 2245
I/Os, 2250
Messaging, 2250
Object name, 2245
Operating modes, 2247
Startup characteristics, 2247
Status word allocation, 2247
Flutter alarm
MonDi08, 822
MonDiL, 779
Flutter suppression, 73, 104
FbAnIn, 2469
FbAnOu, 2478
FbAnTot, 2489
FbDiIn, 2497
FbDiOu, 2507
Pcs7AITC, 2556
Pcs7AnOu, 2581
Pcs7Cnt2, 2640
Pcs7DiIn, 2599
Pcs7DiOu, 2619
Psc7AnIn, 2569
Psc7Cnt1, 2628
Psc7Cnt3, 2648
Psc7DiIT, 2608
FM controller
Preview, 331
Standard view, 294, 298, 302, 306
FmCont
Actuator active information, 884
Anti-windup, 887
Area of application, 876
Associated values, 896
Block diagram, 913
Button labels, 892

Configurable reactions using the Feature parameter, 889
Configurable reactions using the Feature2 parameter, 889
Configuration, 876
Error handling, 892
Error signal generation and dead band, 885
External/internal setpoint specification, 885
Feedforwarding and limiting disturbance variables, 887
Forming the signal status for blocks, 888
Functions, 882
Generating actuating signals for step controllers without position feedback (WithRbk = 0), 884
Generating instance-specific messages, 891
Generation of manipulated variables for controllers, 883
Gradient limit of the setpoint, 885
Group error, 884
How it works, 876
I/Os, 897
Instance-specific messages, 895
Inverting control direction, 886
Limit monitoring of error signal, 886
Limit monitoring of position feedback, 884
Limit monitoring of the process value, 885
Messaging, 894
Module types, 882
Neutral position, 884
Object name, 876
Opening additional faceplates, 892
Operating modes, 880
Operator permissions, 889
Outputting a signal for start readiness, 884
Overview of error numbers, 893
Physical standardization of setpoint, manipulated variable and process value, 886
PID algorithm, 887
Process control fault, 894
Process messages, 895
Release for maintenance, 891
Selecting a unit of measure, 886
Setpoint limiting for external setpoints, 885
SIMATIC BATCH functionality, 892
Simulating signals, 885
Specifying the display area for process and setpoint values as well as operations, 891
Startup characteristics, 877
Status word allocation, 877
Structure segmentation at controllers, 887
Suppressing messages using the MsgLock parameter, 891
Time-stamped messages, 892
Tracking and limiting a manipulated variable, 884
Tracking setpoint in manual mode, 885
Using setpoint ramp, 885
FMCont
Preview, 331
FmTemp
Actuator active information, 924
Anti-windup, 928
Area of application, 916
Associated values, 937
Block diagram, 956
Button labels, 933
Configurable reactions using the Feature parameter, 930
Configurable reactions using the Feature2 parameter, 931
Configuration, 916
Error handling, 934
Error signal generation and dead band, 926
External/internal setpoint specification, 925
Feedforwarding and limiting disturbance variables, 928
Forming the signal status for blocks, 929
Functions, 922
Generating actuating signals for step controllers without position feedback (WithRbk = 0), 924
Generating instance-specific messages, 933
Generation of manipulated variables for controllers, 923
Group error, 924
How it works, 916
I/Os, 938
Instance-specific messages, 937
Inverting control direction, 926
Limit monitoring of error signal, 926
Limit monitoring of position feedback, 925
Limit monitoring of the process value, 925
Limitation of rate of change of setpoint, 925
Messaging, 935
Module types, 922
Neutral position, 924
Object name, 916
Online optimization of the PID controller parameters, 928
Opening additional faceplates, 933
Operating modes, 920
Operator permissions, 931
Outputting a signal for start readiness, 924
Overview of error numbers, 934
Physical standardization of setpoint, manipulated variable and process value, 926

PID algorithm, 927
Process control fault, 935
Process messages, 936
Release for maintenance, 933
Selecting a unit of measure, 927
Setpoint limiting for external setpoints, 925
SIMATIC BATCH functionality, 933
Simulating signals, 925
Specifying the display area for process and setpoint values as well as operations, 933
Startup characteristics, 917
Status word allocation, 917
Structure segmentation at controllers, 927
Suppressing messages using the MsgLock parameter, 933
Time-stamped messages, 933
Tracking and limiting a manipulated variable, 924
Tracking setpoint in manual mode, 925
Using setpoint ramp, 925
FMTemp
Preview, 331
Force operating states
VlvPosL, 1900
Forced tracking in closed-loop controllers, 45
Forcing operating modes
DoseL, 1320
General description, 45
MotL, 1385
MotRevL, 1451
MotSpdCL, 1503
MotSpdL, 1567
Vlv2WayL, 1655
VlvAnL, 1820
VlvL, 1698
VlvMotL, 1768
Formation of the difference between setpoint and position feedback, 1500
Forming a peripheral value
Pcs7DiOu, 2619
Forming an I/O value
Pcs7AnOu, 2580
Forming the group status for interlocks
DoseL, 1321
MotL, 1385
MotRevL, 1450
MotS, 1421
MotSpdCL, 1502
MotSpdL, 1566
Vlv2WayL, 1657
VlvAnL, 1819
VlvL, 1698
VlvMotL, 1767
VlvPosL, 1898
VlvS, 1734
Forming the signal status for blocks
AbsR, 2208
Add04, 2211
Add08, 2216
AV, 708
Average, 2222
CompAn02, 2320
ConPerMon, 854
CountOh, 2105
CountScL, 2081
DeadTime, 2229
Derivative, 2236
Div02, 2241
DoseL, 1321
Event, 2022
EventNck, 2034
FlowCorr, 2248
FmCont, 888
FmTemp, 929
Integral, 2256
Intlk02, 1953
Intlk04, 1968
Intlk08, 1983
Intlk16, 2000
KalFilt, 1284
Lag, 2264
MeanTime, 2271
ModPreCon, 984
MonAnL, 722
MonAnS, 753
MonDi08, 820
MonDiL, 777
MonDiS, 801
MotL, 1385
MotRevL, 1450
MotS, 1421
MotSpdCL, 1502
MotSpdL, 1566
MPC10x10, 1248
Mul04, 2277
Mul08, 2282
MuxAn03, 2334
MuxST, 2771
OpAnL, 391
OpAnS, 414
OpDi01, 429
OpDi03, 444
OpTrig, 472
PIDConL, 1029
PIDConR, 1112

PIDConS, 1076
PIDStepL, 1154
Polygon, 2290
PowerXY, 2315
Ratio, 1200
RedAn02, 2353
RedDi02, 2432
SelA16In, 2365
SelAn16L, 2383
Smooth, 2299
SqrRoot, 2304
Sub02, 2309
TimerP, 2177
TotalL, 2134
Trigger, 2453
Vlv2WayL, 1657
VlvAnL, 1819
VlvL, 1698
VlvMotL, 1767
VlvPosL, 1898
VlvS, 1734
XOr04, 2458
Forming the signal status for PCS 7 channel blocks, 127
Forming the signal status for the block
MuxAn08, 2341
Frequency converter with separate device feed, 162
Feature parameter, 162
Frequency converters, 136
Functions
AbsR, 2208
Add04, 2211
Add08, 2216
And04, 2400
And08, 2405
AssetM, 2781
AV, 707
Average, 2222
CompAn02, 2320
ConPerMon, 845
CountOh, 2103
CountScL, 2080
DeadTime, 2228
Derivative, 2236
Div02, 2241
DoseL, 1312
Event, 2020
Event16Ts, 2062
EventNck, 2033
EventTs, 2046
FbAnIn, 2468
FbAnOu, 2478
FbAnTot, 2488
FbDiIn, 2497
FbDiOu, 2507
FbEnMe, 2530
FbSwtMMS, 2542
FirstIn, 2014
FlipFlop, 2412
FlowCorr, 2247
FmCont, 882
FmTemp, 922
GainSched, 962
Integral, 2255
Intlk02, 1952
Intlk04, 1965
Intlk08, 1980
Intlk16, 1997
KalFilt, 1284
Lag, 2263
Limit, 2326
MeanTime, 2270
ModPreCon, 976
MonAnL, 719
MonAnS, 753
MonDi08, 819
MonDiL, 774
MonDiS, 800
MotL, 1382
MotRevL, 1447
MotS, 1419
MotSpdCL, 1496
MotSpdL, 1563
MPC10x10, 1243
MSTIn, 2739
MSTOu, 2743
Mul04, 2277
Mul08, 2282
MuxAn03, 2331
MuxAn08, 2339
MuxMST, 2766
MuxST, 2771
Not01, 2428
OpAnL, 391
OpAnS, 414
OpDi01, 428
OpDi03, 443
OpStations, 460
OpTrig, 471
Or04, 2419
Or08, 2423
Pcs7AITC, 2554
Pcs7AnIn, 2567
Pcs7AnOu, 2580

Pcs7Cnt2, 2639
Pcs7Cnt3, 2648
Pcs7DiCnt, 2589
Pcs7DiIn, 2598
Pcs7DiIT, 2607
Pcs7DiOu, 2619
Pcs7HaAI, 2666
Pcs7HaAO, 2673
PIDConL, 1024
PIDConR, 1103
PIDConS, 1073
PIDStepL, 1147
Polygon, 2290
PowerXY, 2315
Psc7Cnt1, 2627
RateLim, 2345
Ratio, 1198
Rcv_AnaVal, 2813
Rcv_DigVal, 2807
RedAn02, 2353
RedDi02, 2432
SelA02In, 2358
SelA16In, 2364
SelAn16L, 2382
SelD02In, 2437
ShrdResL, 1607
ShrdResS, 1629
Smooth, 2298
Snd_AnaVal, 2810
Snd_DigVal, 2802
SplRange, 1218
SqrRoot, 2304
STIn, 2731
STOu, 2735
StrctCom, 2442
StrctDeC, 2447
StrgToBy, 2702
StruAnIn, 2707
StruAnOu, 2711
StruDiIn, 2715
StruDiOu, 2719
StruScIn, 2723
StruScOu, 2727
Sub02, 2309
TimerP, 2174
TimeTrig, 2184
TotalL, 2133
Trigger, 2453
Vlv2WayL, 1653
VlvAnL, 1816
VlvL, 1696
VlvMotL, 1762
VlvPosL, 1884
VlvS, 1732
XOr04, 2457
Functions of CntOhSc, 2158
Functions of ReaLToDw, 2748
Functions of STRep, 2776

## G

Gain scheduling, 2825, 2845, 2876
GainSched, 960
GainSched
Area of application, 959
Block diagram, 966
Configurable reactions using the Feature parameter, 962
Configuration, 960
Error handling, 963
Functions, 962
Gain scheduling, 960
How it works, 959
I/Os, 964
Messaging, 963
Object name, 959
Operating modes, 961
Preview, 969
Selecting a unit of measure, 962
Startup characteristics, 961
Status word allocation, 961
General function MV difference
VlvAnL, 1829
General preview
ShrdResL, 1622
Generate instance-specific messages
VlvPosL, 1904
Generating actuating signals for step controllers without position feedback (WithRbk = 0)
FmCont, 884
FmTemp, 924
Generating and limiting the manipulated variable
ModPreCon, 976
MPC10x10, 1243
Generating instance-specific messages, 230
AV, 708
ConPerMon, 856
DoseL, 1323
FmCont, 891
FmTemp, 933
MonAnL, 725
MonAnS, 755
MonDiL, 776
MonDiS, 801

MotL, 1387
MotRevL, 1454
MotS, 1422
MotSpdCL, 1507
MotSpdL, 1571
PIDConL, 1033
PIDConR, 1116
PIDConS, 1079
PIDStepL, 1158
Vlv2WayL, 1658
VlvAnL, 1823
VlvL, 1700
VlvMotL, 1771
VlvS, 1735
Generation of actuating signal without position feedback
PIDStepL, 1148
Generation of manipulated variables
PIDConL, 1024
PIDConR, 1103
PIDConS, 1074
PIDStepL, 1148
VlvAnL, 1828
Generation of manipulated variables for controllers
FmCont, 883
FmTemp, 923
Good state to locked, 58
Gradient limit of the setpoint, 132
Activate, 335
FmCont, 885
MotSpdCL, 1500
OpAnL, 391
PIDConL, 1025
PIDConR, 1107
PIDStepL, 1150
Gradient limiting of the manipulated variable, 134
VlvAnL, 1818
Gradient monitoring
MonAnL, 720
Group display for limit monitoring
CSF and ExtMsgx, 92, 2103
SumMsgAct, 708, 719, 753, 850, 885, 925, 1025, 1106, 1150, 1319, 1499, 1830, 2080, 2103, 2133
Group error, 130
DoseL, 1321
FmCont, 884
FmTemp, 924
MotL, 1384, 1421
MotRevL, 1450
MotSpdCL, 1502
MotSpdL, 1566
PIDConL, 1025
PIDConR, 1105
PIDConS, 1074
PIDStepL, 1150
Vlv2WayL, 1656
VlvAnL, 1817
VlvL, 1697
VlvMotL, 1766
VlvPosL, 1898
Group status
Forming, 113

## H

Handling non-connected inputs
Intlk02, 1952
Intlk04, 1967
Intlk08, 1982
Intlk16, 1999
Harris index, 851
Higher-level error / invalid measuring range
FbAnIn, 2471
FbAnOu, 2480
FbDiIn, 2499
FbDiIT, 2610
FbDiOu, 2509
Pcs7AITC, 2559
Pcs7AnIn, 2572
Pcs7AnOu, 2583
Pcs7DiCnt, 2591
Pcs7DiIn, 2601
Pcs7DiOu, 2620
High-precision time stamp, 231
Hold and restart calculation
Lag, 2264
Hold last value
Pcs7AnIn, 2568
Pcs7DiIn, 2599
Pcs7DiIT, 2608
Holding the last value if raw value is invalid
FbAnIn, 2468
FbAnTot, 2488
FbDiIn, 2497
Pcs7AITC, 2555
Pcs7DiCnt, 2590
How it works
AbsR, 2207
Add04, 2210
Add08, 2215
And04, 2399
And08, 2404
AssetM, 2778

AV, 705
Average, 2220
CompAn02, 2319
ConPerMon, 841
CountOh, 2098
CountScL, 2076
DeadTime, 2226
Derivative, 2233
Div02, 2240
DoseL, 1305
Event, 2017
Event16Ts, 2057
EventNck, 2030
EventTs, 2042
FbAnIn, 2466
FbAnOu, 2476
FbAnTot, 2487
FbDiIn, 2495
FbDiOu, 2505
FbEnMe, 2528
FbSwtMMS, 2541
FirstIn, 2012
FlipFlop, 2409
FlowCorr, 2245
FmCont, 876
FmTemp, 916
GainSched, 959
Integral, 2253
Intlk02, 1949
Intlk04, 1962
Intlk08, 1976
Intlk16, 1992
KalFilt, 1276
Lag, 2261
Limit, 2324
MeanTime, 2268
ModPreCon, 970
MonAnL, 715
MonAnS, 750
MonDi08, 816
MonDiL, 770
MonDiS, 797
MotL, 1376
MotRevL, 1440
MotS, 1415
MotSpdCL, 1488
MotSpdL, 1556
MPC10x10, 1234
MSTIn, 2738
MSTOu, 2742
Mul04, 2275
Mul08, 2281
MuxAn03, 2330
MuxAn08, 2338
MuxMST, 2765
MuxST, 2770
NoiseGen, 2461
Not01, 2427
OpAnL, 389
OpAnS, 412
OpDi01, 426
OpDi03, 441
OpStations, 457
OpTrig, 470
Or04, 2417
Or08, 2422
Pcs7AITC, 2551
Pcs7AnIn, 2564
Pcs7AnOu, 2577
Pcs7Cnt2, 2636
Pcs7Cnt3, 2646
Pcs7DiCnt, 2588
Pcs7DiIn, 2596
Pcs7DiIT, 2605
Pcs7DiOu, 2616
Pcs7HaAI, 2665
Pcs7HaAO, 2672
PIDConL, 1017
PIDConR, 1094
PIDConS, 1069
PIDStepL, 1141
Polygon, 2288
PowerXY, 2313
Psc7Cnt1, 2625
RateLim, 2344
Ratio, 1196
Rcv_AnaVal, 2812
Rcv_DigVal, 2805
RedAn02, 2352
RedDi02, 2431
SelA02In, 2357
SelA16In, 2362
SelAn16L, 2380
SelD02In, 2436
ShrdResL, 1603
ShrdResS, 1626
Smooth, 2297
Snd_AnaVal, 2809
Snd_DigVal, 2801
SplRange, 1216
SqrRoot, 2303
STIn, 2730
STOu, 2734
StrctCom, 2441

StrctDeC, 2446
StruAnIn, 2706
StruAnOu, 2710
StruDiIn, 2714
StruDiOu, 2718
StruScIn, 2722
StruScOu, 2726
StruToBy, 2701
Sub02, 2308
TimerP, 2173
TimeTrig, 2181
TotalL, 2123
Trigger, 2451
Vlv2WayL, 1647
VlvAnL, 1808
VlvL, 1691
VlvMotL, 1754
VlvPosL, 1876
VlvS, 1728
XOr04, 2456
How the block works as RS-FlipFlop (Mode = 1)
FlipFlop, 2410
How the block works as SR-FlipFlop (Mode = 0)
FlipFlop, 2409
Hysteresis, 95, 100, 102, 103, 104, 221, 355

## I

I action, 81, 2836, 2840, 2843, 2874
I component, 887
I/Os
AbsR, 2208
Add04, 2213
Add08, 2218
And04, 2401
And08, 2406
AV, 711
Average, 2224
CntOhSc, 2161
CompAn02, 2322
ConPerMon, 860
CountOh, 2109
CountScL, 2085
DeadTime, 2230
Derivative, 2238
Div02, 2243
DoseL, 1332
Event, 2025
EventNck, 2037
EventTs, 2052, 2071
FbAnIn, 2472
FbAnOu, 2481
FbAnTot, 2491
FbDiIn, 2500
FbDiOu, 2510
FbDrive, 2520
FbEnMe, 2532
FbSwtMMS, 2545
FirstIn, 2015
FlipFlop, 2414
FlowCorr, 2250
FmCont, 897
FmTemp, 938
GainSched, 964
Integral, 2258
Intlk02, 1957
Intlk08, 1985
Intlk16, 2003
Lag, 2266
Limit, 2328
MeanTime, 2273
ModPreCon, 990
MonAnL, 729
MonAnS, 758
MonDi08, 824
MonDiL, 781
MonDiS, 805
MotL, 1394
MotRevL, 1463
MotS, 1426
MotSpdCL, 1518, 1578
MotSpdL, 1578
MSTIn, 2740
MSTOu, 2744
Mul04, 2278
Mul08, 2284
MuxAn03, 2335
MuxAn08, 2341
MuxMST, 2767
MuxST, 2773
NoiseGen, 2461
Not01, 2429
OpAnL, 397
OpAnS, 417
OpDi01, 432
OpDi03, 448
OpStations, 462
OpTrig, 474
Or04, 2420
Or08, 2425
Pcs7AnIn, 2572
Pcs7AnOu, 2584
Pcs7Cnt2, 2642
Pcs7Cnt3, 2650

Pcs7DiIn, 2601
Pcs7DiIT, 2611
Pcs7DiOu, 2621
Pcs7HaAI, 2668
Pcs7HaAO, 2675
PIDConL, 1039
PIDConLS, 1082
PIDConR, 1121
PIDStepL, 1163
Polygon, 2293
PowerXY, 2316
Psc7Cnt1, 2630
RateLim, 2349
Ratio, 1202
RedAn02, 2354
RedDi02, 2433
SelA02In, 2360
SelA16In, 2368
SelAn16L, 2386
SelD02In, 2439, 2454
ShrdResL, 1614
ShrdResS, 1633
Smooth, 2301
SplRange, 1222
SqrRoot, 2306
StateMap, 2751
STIn, 2732
STOu, 2736
StrctCom, 2443
StrctDeC, 2448
StrgToBy, 2703
StruAnIn, 2708
StruAnOu, 2712
StruDiIn, 2716
StruDiOu, 2720
StruScIn, 2724
StruScOu, 2728
Sub02, 2311
TimerP, 2178
TimeTrig, 2192
TotalL, 2140
Vlv2WayL, 1664
VlvAnL, 1835
VlvL, 1707
VlvMotL, 1778
VlvPosL, 1912
VlvS, 1739
XOr04, 2459
I/Os of Rcv_AnaVal, 2814
I/Os of Rcv_DigVal, 2807
I/Os of RealToDw, 2749
I/Os of Snd_AnaVal, 2811
I/Os of Snd_DigVal, 2803
I/Os of STRep, 2776
IMC principle (internal model control), 2830
Import/Export Assistant, 2819
In progress
Operating mode, 70
Increasing availability
MuxAn03, 2332
MuxAn08, 2339
Increasing safety
MuxAn03, 2333
MuxAn08, 2339, 2340
Influence of the signal status on the interlock, 112
Information on areas of application
ModPreCon, 971
MPC10x10, 1235
Initialization
FlipFlop, 2413
Limit, 2327
TimerP, 2176
Input parameter for feedback value
OpDi01, 428
OpDi03, 444
OpTrig, 472
Instance-specific messages, 230, 894, 935, 1036, 1080, 1118, 1160
ConPerMon, 859
DoseL, 1330
FmCont, 895
FmTemp, 937
MonAnL, 728
MonAnS, 757
MotL, 1393
MotRevL, 1462
MotS, 1426
MotSpdCL, 1517
MotSpdL, 1577
PIDConL, 1037
PIDConR, 1120
PIDConS, 1081
PIDStepL, 1162
Vlv2WayL, 1663
VlvL, 1706
VlvMotL, 1777
VlvPosL, 1911
VlvS, 1738
Int64ToDi
Area of application, 2790
Object name, 2790
Integral
Area of application, 2253
Block diagram, 2259

Configurable reactions using the Feature parameter, 2256
Configuration, 2254
Error handling, 2257
Forming the signal status for blocks, 2256
Functions, 2255
How it works, 2253
I/Os, 2258
Messaging, 2258
Monitoring limits, 2255
Object name, 2253
Operating modes, 2254
Overview of error numbers, 2257
Startup characteristics, 2254
Status word allocation, 2254
Stopping integration, 2256
Tracking values, 2255
Interface for the primary controller functions
Description, 85
Interlock block
Activating recording of the first signal, 161
Block icon, 272
Preview, 333
Recording the first signal, 58
Standard view, 310
Interlock blocks, 2856, 2857, 2858, 2859, 2860, 2861, 2862, 2863, 2864, 2865, 2866, 2867
Interlocks
DoseL, 1320
MotL, 1383
MotRevL, 1449
MotS, 1420
MotSpdCL, 1501
MotSpdL, 1565
OpDi01, 428
OpDi03, 443
PIDConL, 1033, 1116
Vlv2WayL, 1656
VlvAnL, 1817
VlvL, 1697
VlvMotL, 1764
VlvPosL, 1894
VlvS, 1733
Interlocks at blocks, 107
Internal or external digital value
OpDi01, 428
OpDi03, 443
Internal or external ratio
Ratio, 1199
Internal or external setpoint selection
OpAnL, 391
OpAnS, 414
Internal simulation, 65
IntIk04
Configurable reactions using the Feature I/O, 1969
Intlk016
Bypass, 1998
Intlk02
Area of application, 1949
Block diagram, 1961
Bypass, 1953
Configurable reactions using the Feature parameter, 1956
Configuration, 1949
Error handling, 1956
First-in detection, 1953
Forming the signal status for blocks, 1953
Functions, 1952
Handling non-connected inputs, 1952
How it works, 1949
I/Os, 1957
Installation in OBs, 1962
Messaging, 1957
Modes, 1952
Object name, 1949
Opening additional faceplates, 1953
Operating permissions, 1955
Overview of error numbers, 1957
Preview, 333
Standard view, 310
Startup characteristics, 1950
Status word allocation, 1950
Intlk04
Area of application, 1962
Block diagram, 1974
Bypass, 1966
Configuration, 1962
Error handling, 1969
First-in detection, 1968
Forming the signal status for blocks, 1968
Functions, 1965
Handling non-connected inputs, 1967
How it works, 1962
I/Os, 1970
Inversion of logic signals, 1965
Messaging, 1970
Modes, 1965
Object name, 1962
Opening additional faceplates, 1968
Operating permissions, 1968
Overview of error numbers, 1970
Preview, 333
Standard view, 310

Startup characteristics, 1962
Status word allocation, 1962
Intlk08
Area of application, 1976
Block diagram, 1991
Bypass, 1980
Configurable reactions using the Feature parameter, 1984
Configuration, 1976
Error handling, 1984
First-in detection, 1983
Forming the signal status for blocks, 1983
Functions, 1980
Handling non-connected inputs, 1982
How it works, 1976
I/Os, 1985
Inversion of logic signals, 1980
Logic operators, 1980
Messaging, 1985
Object name, 1976
Opening additional faceplates, 1982
Operating modes, 1979
Operating permissions, 1983
Overview of error numbers, 1985
Preview, 333
Standard view, 310
Startup characteristics, 1976
Status word allocation, 1976
Intlk16
Area of application, 1992
Block diagram, 2010
Configurable reactions using the Feature parameter, 2001
Configuration, 1992
Error handling, 2002
First-in detection, 2000
Forming the signal status for blocks, 2000
Functions, 1997
Handling non-connected inputs, 1999
How it works, 1992
I/Os, 2003
Inversion of logic signals, 1997
Logic operators, 1997
Messaging, 2002
Modes, 1997
Object name, 1992
Opening additional faceplates, 2000
Operating permissions, 2000
Overview of error numbers, 2002
Preview, 333
Standard view, 310
Startup characteristics, 1992
Status word allocation, 1992
Invalid input signals
MotS, 1424
VlvAnL, 1832
VlvS, 1737
Invalid or non plausible input commands
MotL, 1391
MotSpdCL, 1513
Invalid or not plausible input commands
MotRevL, 1458
MotSpdL, 1575
Vlv2WayL, 1661
VlvL, 1704
VlvMotL, 1775
VlvPosL, 1909
Invalid raw value, 161, 165, 185
Invalid value
Pcs7HaAI, 2667
Pcs7HaAO, 2674
Inversion of logic signals
Intlk04, 1965
Intlk08, 1980
Intlk16, 1997
Inverting control direction, 217
FmCont, 886
FmTemp, 926
PIDConL, 1027
PIDConR, 1108
PIDStepL, 1151
Issuing a request for maintenance (MS → APL)
Request for maintenance release, 68
Issuing trigger signal internally or externally
OpTrig, 472


## K


KalFilt
Area of application, 1276
Block diagram, 1292
Block icon, 1302
Configurable reactions using the Feature parameter, 1284
Configuration, 1279
Error handling, 1286
Forming the signal status for blocks, 1284
Functions, 1284
How it works, 1276
Kalman Configurator, 1279
Measurements view, 1301
Messaging, 1292
Object name, 1276

Operating modes, 1282
Operating principle, 1278
Operator permissions, 1285
Overview of error numbers, 1286
Parameter view, 1297
Parameter view 2, 1298
Preview, 1299
Standard view, 1293
Startup characteristics, 1281
Status word allocation, 1281
Trend view, 1302
Kalman Configurator
KalFilt, 1279
Keep last valid raw value
FbAnOu, 2479
FbDiOu, 2508
Pcs7AnOu, 2581
Pcs7DiOu, 2619


## L


Labeling of buttons
VlvPosL, 1907
Lag
Area of application, 2261
Block diagram, 2267
Configurable reactions using the Feature parameter, 2264
Configuration, 2262
Error handling, 2264
Forming the signal status for blocks, 2264
Functions, 2263
Hold and restart calculation, 2264
How it works, 2261
I/Os, 2266
Messaging, 2265
Object name, 2261
Operating modes, 2263
Overview of error numbers, 2265
Reset values, 2264
Startup characteristics, 2263
Status word allocation, 2263
Limit
Area of application, 2324
Block diagram, 2329
Configuration, 2325
Error handling, 2327
Functions, 2326
How it works, 2324
I/Os, 2328
Initialization, 2327
Messaging, 2328
Object name, 2324
Operating modes, 2326
Startup characteristics, 2326
Status word allocation, 2326
Limit monitoring
Derivative, 2236
Error signal, 102
Manipulated variable difference, 102
Setpoint difference, 102
Limit monitoring for additional analog value
VlvPosL, 1893
Limit monitoring of an additional analog value
AV, 708
MotL, 1383
MotRevL, 1448
MotSpdCL, 1499
MotSpdL, 1564
VlvMotL, 1764
Limit monitoring of error signal
FmCont, 886
FmTemp, 926
PIDConL, 1027
PIDConR, 1108
PIDStepL, 1151
Limit monitoring of manipulated variable and error signal
VlvAnL, 1830
Limit monitoring of position feedback
FmCont, 884
FmTemp, 925
PIDConL, 1025
PIDConR, 1106
PIDStepL, 1150
Limit monitoring of the count value
CountScL, 2080
TotalL, 2133
Limit monitoring of the feedback
MotSpdCL, 1499
Limit monitoring of the operating time
CountOh, 2103
Limit monitoring of the process value
DoseL, 1319
FmCont, 885
FmTemp, 925
MonAnL, 719
MonAnS, 753
PIDConL, 1026
PIDConR, 1107
PIDConS, 1075
PIDStepL, 1151
Limit monitoring of the setpoint difference
MotSpdCL, 1501

Limit monitoring with hysteresis
AV, 708
MotL, 1383
MotRevL, 1448
MotSpdCL, 1499
MotSpdL, 1564
VlvMotL, 1764
VlvPosL, 1893
Limit operation and display in the faceplate
ConPerMon, 856
Limit view
DoseL, 1354
FM controller, 322
Motor, 328
MotSpdCL, 1546
PID controller, 325
Limit view of CntOhSc, 2168
Limit violation, 128
Limitation of rate of change of setpoint
FmTemp, 925
Limitation of the slope of an analog signal
RateLim, 2346
Limiting the output value
Ratio, 1199
Limiting the peripheral value
Pcs7AnOu, 2581
Limiting the process value
Pcs7AnOu, 2581
Limiting the ratio
Ratio, 1199
Local mode, 184
Logic operators
Intlk08, 1980
Intlk16, 1997
Low pass filter, 847
Low-pass filter, 2877

## M

Main entry
Subentry, 1876
Manipulated variable, 82
Forced tracking, 223
Tracking, 222
Manipulated variable difference, 102
Manipulated variable difference generation and dead band
VlvAnL, 1829
Manipulated variable ramp
Using, 133
Manipulated variable specification
External, 137
Internal, 137
Manipulated variables, 2880
Manual mode
Controller blocks, 79
Dosers, 82
Motors, 82
Valves, 82
MeanTime
Area of application, 2268
Block diagram, 2274
Configurable reactions using the Feature parameter, 2270
Configuration, 2269
Error handling, 2272
Forming the signal status for blocks, 2271
Functions, 2270
How it works, 2268
I/Os, 2273
Messaging, 2272
Object name, 2268
Operating modes, 2269
Overview of error numbers, 2272
Setting a mean value constant, 2270
Startup characteristics, 2269
Status word allocation, 2269
Stopping the calculation of mean values, 2270
Memo view
Description, 338
MemR256
Area of application, 2791
Description, 2791
Description of, 2791
Message characteristics
VlvPosL, 1910
Message classes
User-configured, 45
Message view, 336
Messages
Generating instance-specific messages, 230
Messaging
AbsR, 2208
Add04, 2212
Add08, 2217
And04, 2401
And08, 2406
AV, 710
Average, 2223
CompAn02, 2322
ConPerMon, 857
CountOh, 2108

CountScL, 2084
DeadTime, 2230
Derivative, 2237
Div02, 2243
DoseL, 1329
Event, 2023
Event16Ts, 2065
EventNck, 2036
EventTs, 2049
FbAnIn, 2472
FbAnOu, 2480
FbAnTot, 2491
FbDiIn, 2500
FbDiOu, 2510
FbEnMe, 2532
FbSwtMMS, 2544
FirstIn, 2015
FlipFlop, 2414
FlowCorr, 2250
FmCont, 894
FmTemp, 935
GainSched, 963
Integral, 2258
Intlk02, 1957
Intlk04, 1970
Intlk08, 1985
Intlk16, 2002
KalFilt, 1292
Lag, 2265
Limit, 2328
MeanTime, 2272
ModPreCon, 990
MonAnL, 727
MonAnS, 756
MonDi08, 823
MonDiL, 780
MonDiS, 804
MotL, 1392
MotRevL, 1461
MotS, 1425
MotSpdCL, 1516
MotSpdL, 1577
MPC_10x10, 1254
MSTIn, 2740
MSTOu, 2744
Mul04, 2278
Mul08, 2284
MuxAn03, 2335
MuxAn08, 2341
MuxMST, 2767
MuxST, 2772
Not01, 2429
OpAnL, 395
OpAnS, 417
OpDi01, 431
OpDi03, 447
OpStations, 461
OpTrig, 474
Or04, 2420
Or08, 2424
Pcs7AnIn, 2572
Pcs7AnOu, 2583
Pcs7Cnt2, 2642
Pcs7DiCnt, 2591
Pcs7DiIn, 2601
Pcs7DiIT, 2610
Pcs7DiOu, 2621
Pcs7HaAI, 2668
Pcs7HaAO, 2675
PcsCnt3, 2650
PIDConL, 1036
PIDConR, 1118
PIDConS, 1080
PIDStepL, 1160
Polygon, 2292
PowerXY, 2316
Psc7Cnt1, 2630
RateLim, 2348
Ratio, 1202
Rcv_AnaVal, 2814
Rcv_DigVal, 2807
RedAn02, 2354
RedDi02, 2433
SelA02In, 2360
SelA16In, 2368
SelAn16L, 2386
SelD02In, 2438
ShrdResL, 1613
ShrdResS, 1632
Smooth, 2300
Snd_AnaVal, 2811
Snd_DigVal, 2803
SplRange, 1221
SqrRoot, 2305
StateMap, 2751
STIn, 2732
STOu, 2736
StrctCom, 2443
StrctDeC, 2448
StrgToBy, 2703
StruAnIn, 2708
StruAnOu, 2712
StruDiIn, 2716
StruDiOu, 2720

StruScIn, 2724
StruScOu, 2728
Sub02, 2310
TimerP, 2178
TimeTrig, 2192
TotalL, 2139
Trigger, 2454
Vlv2WayL, 1663
VlvAnL, 1833
VlvL, 1705
VlvMotL, 1777
VlvS, 1738
XOr04, 2458
Messaging of RealToDw, 2748
Messaging of STRep, 2776
mode, 2687, 2698
MODE settings for PA/FF field devices, 2698
Mode Settings for SM Modules, 2688
Mode switchover error
AV, 710
DoseL, 1328
MotL, 1391
MotRevL, 1458
MotS, 1424
MotSpdCL, 1513
MotSpdL, 1575
Vlv2WayL, 1661
VlvAnL, 1832
VlvL, 1704
VlvMotL, 1775
VlvPosL, 1908
VlvS, 1737
Model performance, 2830
Model-based disturbance compensation
ModPreCon, 980
MPC10x10, 1246
Modes
FbAnOu, 2478
FbDiOu, 2506
Intlk02, 1952
Intlk04, 1965
Intlk08, 1979
Intlk16, 1997
Pcs7AnIn, 2566
Pcs7AnOu, 2580
Pcs7DiIn, 2598
Pcs7DiIT, 2607
Pcs7DiOu, 2618
StruAnIn, 2707
StruAnOu, 2711
StruDiIn, 2715
StruDiOu, 2719
StruScIn, 2723
ModPreCon
Anti-windup, 979
Area of application, 970
Block diagram, 1001
Block icon, 1014
Configurable reactions using the Feature parameter, 984
Configuration, 972
Control of linear and non-linear systems, 982
Control of square and non-square systems, 981
Error handling, 989
Error signal generation and dead band, 978
Forming the signal status for blocks, 984
Functions, 976
Generating and limiting the manipulated variable, 976
How it works, 970
I/Os, 990
Information on areas of application, 971
Messaging, 990
Model-based disturbance compensation, 980
Object name, 970
Opening additional faceplates, 986
Operating modes, 975
Operator permissions, 985
Overview of error numbers, 989
Parameter view, 1008
Predictive controller algorithm, 979
Release for maintenance, 986
Selecting a unit of measure, 978
Setpoint filters, 977
Setpoint tracking in manual mode, 977
Setting the setpoint internally, 977
SIMATIC BATCH functionality, 986
Simulating signals, 978
Specifying the display area for process and setpoint values as well as operations, 986
Standard view, 1003
Startup characteristics, 973
Status word allocation, 974
Tracking and limiting a manipulated variable, 977
Module types
FmCont, 882
FmTemp, 922
MonAnL
Area of application, 715
Associated values, 728
Block diagram, 738
Block symbol, 748
Bypass function, 723

Configurable reactions using the Feature parameter, 723
Configuration, 715
Control system fault (CSF), 726, 779
Dead band, 722
Disabling of limits, 723
Displaying auxiliary values, 721
Error handling, 726
Forming the signal status for blocks, 722
Functions, 719
Generating instance-specific messages, 725
Gradient monitoring, 720
How it works, 715
I/Os, 729
Instance-specific messages, 728
Limit monitoring of the process value, 719
Limit view, 743
Messaging, 727
Object name, 715
Opening additional faceplates, 725
Operating modes, 718
Operator permissions, 724
Overview of error numbers, 726, 779
Parameter view, 745
Preview, 746
Process control fault, 727
Process messages, 727
Release for maintenance, 723
Selecting a unit of measure, 723
SIMATIC BATCH functionality, 725
Simulating signals, 723
Smoothing PV, 722
Specifying the display area for process and setpoint values as well as operations, 725
Standard view, 740
Startup characteristics, 715
Status word allocation, 716
Suppressing messages using the MsgLock parameter, 719
Time-stamped messages, 725
MonAnS
Alarm delays with one time value per limit pair, 753
Area of application, 750
Associated values, 757
Block diagram, 762
Block symbol, 768
Configurable reactions using the Feature parameter, 754
Configuration, 750
Control system fault (CSF), 756
Dead band, 753
Error handling, 755
Forming the signal status for blocks, 753
Functions, 753
Generating instance-specific messages, 755
How it works, 750
I/Os, 758
Instance-specific messages, 757
Limit monitoring of the process value, 753
Limit view, 765
Messaging, 756
Object name, 750
Opening additional faceplates, 755
Operating modes, 752
Operator permissions, 754
Overview of error numbers, 756
Parameter view, 766
Preview, 767
Process control fault, 756
Process messages, 757
Release for maintenance, 753
Selecting a unit of measure, 754
SIMATIC BATCH functionality, 755
Simulating signals, 753
Specifying the display area for process and setpoint values as well as operations, 755
Startup characteristics, 750
Status word allocation, 750
Suppressing messages using the MsgLock parameter, 753
MonDi08
Area of application, 816
Associated values, 824
Block diagram, 829
Block symbol, 834
Changing labels on buttons and text, 821
Configurable reactions using the Feature parameter, 820
Configuration, 816
Error handling, 822
Flutter alarm, 822
Forming the signal status for blocks, 820
Functions, 819
How it works, 816
I/Os, 824
Messaging, 823
Monitoring and output of digital signals, 819
Object name, 816
Opening additional faceplates, 821
Operating modes, 818
Operator control permissions, 820
Overview of error numbers, 822
Parameter view, 832

Preview, 833
Process messages, 823
Release for maintenance, 820
SIMATIC BATCH functionality, 821
Simulating signals, 820
Standard view, 830
Startup characteristics, 816
Status word allocation, 816
Suppressing messages using the MsgLock parameter, 820
Time-stamped messages, 821
MonDiL
Adapting the color representation in the configured message class, 775
Area of application, 770
Associated values, 781
Block diagram, 786
Block icon, 793, 813
Bypass function, 777
Changing labels on buttons and text, 776
Configurable reactions using the Feature parameter, 777
Configuration, 771
Delaying the on and off switching functions, 775
Displaying auxiliary values, 776
Error handling, 779
Flutter alarm, 779
Forming the signal status for blocks, 777
Functions, 774
Generating instance-specific messages, 776
How it works, 770
I/Os, 781
Messaging, 780
Object name, 770
Opening additional faceplates, 778
Operating modes, 773
Operator permissions, 777
Parameter view, 790
Preview, 792
Process control fault, 780
Process messages, 780
Release for maintenance, 777
SIMATIC BATCH functionality, 778
Simulating signals, 777
Standard view, 788
Startup characteristics, 771
Status word allocation, 771
Suppressing messages using the MsgLock parameter, 777
Suppression and reporting of signal flutter, 774
Time-stamped messages, 778
MonDiS
Adapting the color representation in the configured message class, 800
Area of application, 796
Associated values, 805
Block diagram, 808
Changing labels on buttons and text, 801
Configurable reactions using the Feature parameter, 802
Configuration, 797
Control system fault (CSF), 803
Delaying the on function, 800
Error handling, 803
Forming the signal status for blocks, 801
Functions, 800
Generating instance-specific messages, 801
How it works, 797
I/Os, 805
Messaging, 804
Object name, 796
Opening additional faceplates, 803
Operating modes, 799
Operator permissions, 802
Overview of error numbers, 803
Parameter view, 811
Preview, 812
Process control fault, 804
Process messages, 804
Release for maintenance, 802
SIMATIC BATCH functionality, 803
Simulating signals, 802
Standard view, 809
Startup characteristics, 797
Status word allocation, 797
Suppressing messages using the MsgLock parameter, 801
Monitoring and output of digital signals
MonDi08, 819
Monitoring error, 48
Monitoring limits
Integral, 2255
Monitoring of deterministic characteristics of the control performance, 848
Monitoring of stochastic characteristics of the control performance, 846
Monitoring the feedback for the auxiliary valve
VlvAnL, 1822
MotL
Area of application, 1376
Associated values, 1393
Block diagram, 1404
Block symbol, 1412

Button labels, 1390, 1420
Configurable reactions using the Feature parameter, 1388
Configuration, 1376
Considering bad quality of automatic commands or external values, 1385
Disabling interlocks, 1384
Displaying auxiliary values, 1390
Error handling, 1390
Feedback monitoring, 1386
Forcing operating modes, 1385
Forming the group status for interlocks, 1385
Forming the signal status for blocks, 1385
Functions, 1382
Generating instance-specific messages, 1387
Group error, 1384, 1421
How it works, 1376
I/Os, 1394
Instance-specific messages, 1393
Interlocks, 1383
Invalid or non plausible input commands, 1391
Limit monitoring of an additional analog value, 1383
Limit monitoring with hysteresis, 1383
Messaging, 1392
Mode switchover error, 1391
Motor protection function, 1384
Neutral position, 1387
Object name, 1376
Opening additional faceplates, 1382
Operating modes, 1380
Operator permissions, 1382
Output signal as a pulse signal or static signal, 1387
Outputting a signal for start readiness, 1384, 1422, 1502
Overview of error numbers, 1391
Preview, 1408
Process control fault, 1392
Rapid stop, 1384
Release for maintenance, 1386, 1423
Resetting the block in case of interlocks or errors, 1384
Selecting a unit of measure, 1387
SIMATIC BATCH functionality, 1390
Simulating signals, 1387
Specify warning times for control functions, 1387
Standard view, 1405
Startup characteristics, 1376
Status word allocation, 1376
Suppressing messages using the MsgLock parameter, 1383, 1420
Time delay after restart, 1385
Time-stamped messages, 1390
Motor
Warning times, 56
Motor protection function, 106
MotL, 1384
MotRevL, 1449
MotS, 1421
MotSpdCL, 1501
MotSpdL, 1565
VlvMotL, 1764
VlvPosL, 1895
MotRevL
Area of application, 1440
Associated values, 1462
Block diagram, 1474
Block icon, 1485
Button labels, 1457
Configurable reactions using the Feature parameter, 1455
Configurable reactions using the Feature2 parameter, 1456
Configuration, 1440
Considering bad quality of automatic commands or external values, 1451
Disabling interlocks, 1449
Displaying auxiliary values, 1457
Error handling, 1457
Feedback monitoring, 1451
Forcing operating modes, 1451
Forming the group status for interlocks, 1450
Forming the signal status for blocks, 1450
Functions, 1447
Generating instance-specific messages, 1454
Group error, 1450
How it works, 1440
I/Os, 1463
Instance-specific messages, 1462
Interlocks, 1449
Invalid or not plausible input commands, 1458
Limit monitoring of an additional analog value, 1448
Limit monitoring with hysteresis, 1448
Messaging, 1461
Mode switchover error, 1458
Motor protection function, 1449
Neutral position, 1454
Object name, 1440
Opening additional faceplates, 1447
Operating modes, 1445
Operator permissions, 1447

Output signal as a pulse signal or static signal, 1454
Outputting a signal for start readiness, 1450
Overview of error numbers, 1458
Preview, 1479
Process control fault, 1461
Rapid stop, 1449
Release for maintenance, 1453
Resetting the block in case of interlocks or errors, 1449
Selecting a unit of measure, 1454
SIMATIC BATCH functionality, 1457
Simulating signals, 1454
Specify warning times for control functions, 1453
Standard view, 1475
Startup characteristics, 1440
Status word allocation, 1440
Suppressing messages using the MsgLock parameter, 1449
Time delay after changing direction of rotation or restart, 1448
Time-stamped messages, 1457
MotS
Area of application, 1415
Associated values, 1426
Block diagram, 1432
Block symbol, 1438
Configurable reactions using the Feature parameter, 1422
Configuration, 1415
Considering bad quality of automatic commands or external values, 1422
Disabling interlocks, 1421
Error handling, 1424
Feedback monitoring, 1422
Forming the group status for interlocks, 1421
Forming the signal status for blocks, 1421
Functions, 1419
Generating instance-specific messages, 1422
How it works, 1415
I/Os, 1426
Instance-specific messages, 1426
Interlocks, 1420
Invalid input signals, 1424
Messaging, 1425
Mode switchover error, 1424
Motor protection function, 1421
Neutral position, 1422
Object name, 1415
Opening additional faceplates, 1419
Operating modes, 1418
Operator permissions, 1419
Output signal as a pulse signal or static signal, 1422
Overview of error numbers, 1424
Preview, 1436
Process control fault, 1425
Resetting the block in case of interlocks or errors, 1421
SIMATIC BATCH functionality, 1423
Simulating signals, 1422
Standard view, 1433
Startup characteristics, 1415
Status word allocation, 1415
MotSpdCL
Alarm delays with one time value per limit pair, 1496
Area of application, 1488
Associated values, 1518
Block diagram, 1535
Block icon, 1552
Button labels, 1510
Configurable reactions using the Feature parameter, 1508
Configurable reactions using the Feature2 parameter, 1509
Configuration, 1488
Considering bad quality of automatic commands or external values, 1503
Controlling a device infeed or an inverter enable, 1510
Disabling interlocks, 1502
Disabling of limits, 1507
Displaying auxiliary values, 1510
Error handling, 1512
External/internal setpoint specification, 1499
Feedback monitoring, 1504
Forcing operating modes, 1503
Forming the group status for interlocks, 1502
Forming the signal status for blocks, 1502
Functions, 1496
Generating instance-specific messages, 1507
Gradient limit of the setpoint, 1500
Group error, 1502
How it works, 1488
I/Os, 1518
Instance-specific messages, 1517
Interlocks, 1501
Invalid or non plausible input commands, 1513
Limit monitoring of an additional analog value, 1499
Limit monitoring of the feedback, 1499
Limit monitoring of the setpoint difference, 1501
Limit monitoring with hysteresis, 1499

Messaging, 1516
Mode switchover error, 1513
Motor protection function, 1501
Neutral position, 1506
Object name, 1488
Opening additional faceplates, 1496
Operating modes, 1494
Operator permissions, 1497
Output signal as a pulse signal or static signal, 1507
Overview of error numbers, 1512
Parameter view, 1548
Preview, 1542
Process control fault, 1516
Rapid stop, 1502
Release for maintenance, 1506
Resetting the block in case of interlocks or errors, 1502
Selecting a unit of measure, 1506
Setpoint limitation, 1499
Setpoint ramp, 1500
SIMATIC BATCH functionality, 1510
Simulating signals, 1506
Specify warning times for control functions, 1506
Specifying the display and operator input area for process values and setpoints, 1506
Standard view, 1536
Startup characteristics, 1488
Status word allocation, 1489
Subentry, 1499
Suppressing messages using the MsgLock parameter, 1499
Time delay after changing direction of rotation or restart, 1498
Time-stamped messages, 1510
MotSpdL
Area of application, 1556
Associated values, 1578
Block diagram, 1590
Block icon, 1600
Button labels, 1574
Configurable reactions using the Feature parameter, 1572
Configurable reactions using the Feature2 parameter, 1573
Configuration, 1556
Considering bad quality of automatic commands or external values, 1567
Disabling interlocks, 1565
Displaying auxiliary values, 1574
Error handling, 1574
Feedback monitoring, 1567
Forcing operating modes, 1567
Forming the group status for interlocks, 1566
Forming the signal status for blocks, 1566
Functions, 1563
Generating instance-specific messages, 1571
Group error, 1566
How it works, 1556
I/Os, 1578
Instance-specific messages, 1577
Interlocks, 1565
Invalid or not plausible input commands, 1575
Limit monitoring of an additional analog value, 1564
Limit monitoring with hysteresis, 1564
Messaging, 1577
Mode switchover error, 1575
Motor protection function, 1565
Neutral position, 1571
Object name, 1556
Opening additional faceplates, 1563
Operating modes, 1561
Operator permissions, 1563
Output signal as a pulse signal or static signal, 1571
Outputting a signal for start readiness, 1566
Overview of error numbers, 1575
Preview, 1595
Process control fault, 1577
Rapid stop, 1565
Release for maintenance, 1570
Resetting the block in case of interlocks or errors, 1565
Selecting a unit of measure, 1571
SIMATIC BATCH functionality, 1574
Simulating signals, 1570
Specify warning times for control functions, 1570
Startup characteristics, 1556
Status word allocation, 1556
Step control mode for the speed change, 1570
Suppressing messages using the MsgLock parameter, 1565
Time delay after restart, 1564
Time-stamped messages, 1574
MPC Configurator, 977, 1244
MPC_10x10
Messaging, 1254
MPC10x10
Anti-windup, 1246
Area of application, 1234
Block diagram, 1260
Block icon, 1274

Configurable reactions using the Feature parameter, 1249
Configuration, 1237
Control of linear and non-linear systems, 1248
Control of square and non-square systems, 1247
Error handling, 1253
Error signal generation and dead band, 1245
Forming the signal status for blocks, 1248
Functions, 1243
Generating and limiting the manipulated variable, 1243
How it works, 1234
Information on areas of application, 1235
Model-based disturbance compensation, 1246
Object name, 1234
Opening additional faceplates, 1251
Operating modes, 1241
Operator permissions, 1249
Overview of error numbers, 1254
Parameter view, 1267
Predictive controller algorithm, 1245
Release for maintenance, 1250
Selecting a unit of measure, 1244
Setpoint filters, 1244
Setpoint tracking in manual mode, 1244
Setting the setpoint internally, 1243
SIMATIC BATCH functionality, 1251
Simulating signals, 1244
Specifying the display area for process and setpoint values as well as operations, 1250
Standard view, 1261
Startup characteristics, 1240
Status word allocation, 1240
Tracking and limiting a manipulated variable, 1243
MSTIn
Area of application, 2738
Block diagram, 2741
Configuration, 2738
Error handling, 2739
Functions, 2739
How it works, 2738
I/Os, 2740
Messaging, 2740
Object name, 2738
Operating modes, 2739
Startup characteristics, 2738
Status word allocation, 2738
MSTOu
Area of application, 2742
Block diagram, 2745
Configuration, 2742
Error handling, 2744
Functions, 2743
How it works, 2742
I/Os, 2744
Messaging, 2744
Object name, 2742
Operating modes, 2743
Startup characteristics, 2742
Status word allocation, 2742
Mul04
Area of application, 2275
Block diagram, 2279
Configuration, 2275
Error handling, 2278
Forming the signal status for blocks, 2277
Functions, 2277
How it works, 2275
I/Os, 2278
Messaging, 2278
Object name, 2275
Operating modes, 2276
Startup characteristics, 2276
Status word allocation, 2276
Mul08
Area of application, 2281
Block diagram, 2285
Configuration, 2281
Error handling, 2283
Forming the signal status for blocks, 2282
Functions, 2282
How it works, 2281
I/Os, 2284
Messaging, 2284
Object name, 2281
Operating modes, 2282
Startup characteristics, 2281
Status word allocation, 2282
Multi-model controlling, 2879
Multivariable controller
ConPerMon, 853
Definition, 2888
MuxAn03
Area of application, 2330
Block diagram, 2337
Configuration, 2330
Error handling, 2334
Forming the signal status for blocks, 2334
Functions, 2331
How it works, 2330
I/Os, 2335
Increasing availability, 2332
Increasing safety, 2333

Messaging, 2335
Object name, 2330
Operating modes, 2331
Selection of output signal, 2331
Startup characteristics, 2330
Status word allocation, 2330
MuxAn08
Area of application, 2338
Block diagram, 2343
Configuration, 2338
Error handling, 2341
Forming the signal status for blocks, 2341
Functions, 2339
How it works, 2338
I/Os, 2341
Increasing availability, 2339
Increasing safety, 2340
Messaging, 2341
Object name, 2338
Operating modes, 2338
Selection of output signal, 2339
Startup characteristics, 2338
Status word allocation, 2338
MuxMST
Area of application, 2765
Block diagram, 2768
Configuration, 2765
Error handling, 2767
Functions, 2766
How it works, 2765
I/Os, 2767
Messaging, 2767
Object name, 2765
Operating modes, 2766
Startup characteristics, 2765
Status word allocation, 2765
MuxST
Area of application, 2770
Block diagram, 2774
Configuration, 2770
Error handling, 2772
Forming the signal status for blocks, 2771
Functions, 2771
How it works, 2770
I/Os, 2773
Messaging, 2772
Object name, 2770
Operating modes, 2771
Selecting signals for processing, 2771
Startup characteristics, 2770
Status word allocation, 2770

## N

NegInt64
Area of application, 2792
Object name, 2792
NegR64
Area of application, 2793
Object name, 2793
Neutral position, 54
FmCont, 884
FmTemp, 924
MotL, 1387
MotRevL, 1454
MotS, 1422
MotSpdCL, 1506
MotSpdL, 1571
PIDConL, 1024
PIDConR, 1105
PIDConS, 1074
PIDStepL, 1149
Vlv2WayL, 1654
VlvAnL, 1823
VlvL, 1700
VlvMotL, 1770
VlvPosL, 1903
Noise generator block, 2873
NoiseGen
Area of application, 2461
How it works, 2461
I/Os, 2461
Object name, 2461
Not bumpless, 189
Not01
Area of application, 2427
Block diagram, 2430
Configuration, 2427
Error handling, 2428
Functions, 2428
How it works, 2427
I/Os, 2429
Messaging, 2429
Object name, 2427
Operating modes, 2428

## O

Object name
AbsR, 2207
Add04, 2210
Add08, 2215

AddInt64, 2787
AddR64, 2788
And04, 2399
And08, 2404
Average, 2220
CompAn02, 2319
ConPerMon, 841
CountOh, 2098
CountScL, 2075
DeadTime, 2226
Derivative, 2233
DiToInt64, 2789
Div02, 2240
DoseL, 1305
Event, 2017
Event16Ts, 2057
EventNck, 2030
EventTs, 2042
FbAnIn, 2466
FbAnOu, 2476
FbAnTot, 2487
FbDiIn, 2495
FbDiOu, 2505
FbEnMe, 2528
FirstIn, 2012
FlipFlop, 2409
FlowCorr, 2245
FmCont, 876
FmTemp, 916
GainSched, 959
Int64ToDi, 2790
Integral, 2253
Intlk02, 1949
Intlk04, 1962
Intlk08, 1976
Intlk16, 1992
KalFilt, 1276
Lag, 2261
Limit, 2324
MeanTime, 2268
ModPreCon, 970
MonAnL, 715
MonAnS, 750
MonDi08, 816
MonDiL, 770
MonDiS, 796
MotRevL, 1440
MotS, 1415
MotSpdCL, 1488
MotSpdL, 1556
MPC10x10, 1234
MSTIn, 2738
MSTOu, 2742
Mul04, 2275
Mul08, 2281
MuxAn03, 2330
MuxAn08, 2338
MuxMST, 2765
MuxST, 2770
NegInt64, 2792
NegR64, 2793
NoiseGen, 2461
Not01, 2427
OpAnL, 389
OpAnS, 412
OpDi01, 426
OpDi03, 441
OpStations, 457
OpTrig, 470
Or04, 2417
Or08, 2422
Pcs7AITC, 2551
Pcs7AnIn, 2564
Pcs7AnOu, 2577
Pcs7DiCnt, 2588
Pcs7DiIn, 2596
Pcs7DiIT, 2605
Pcs7DiOu, 2616
Pcs7HaAI, 2665
Pcs7HaAO, 2672
PIDCoefR, 2794
PIDConL, 1017
PIDConR, 1094
PIDConS, 1069
PIDKernR, 2800
PIDStepL, 1141
Polygon, 2287
R64ToReal, 2795
RateLim, 2344
Ratio, 1196
Rcv_AnaVal, 2812
Rcv_DigVal, 2805
RealToR64, 2796
RedAn02, 2352
RedDi02, 2431
SelA02In, 2357
SelA16In, 2362
SelAn16L, 2380
SelD02In, 2436
SelST16, 2797
ShLeInt64, 2798
ShrdResL, 1603
ShrdResS, 1626
ShRiInt64, 2799

Smooth, 2297
Snd_AnaVal, 2809
Snd_DigVal, 2801
SplRange, 1216
SqrRoot, 2303
STIn, 2730
STOu, 2734
StrctCom, 2441
StrctDeC, 2446
StruAnIn, 2706
StruAnOu, 2710
StruDiIn, 2714
StruDiOu, 2718
StruScIn, 2722
StruScOu, 2726
StruToBy, 2701
Sub02, 2308
TimerP, 2173
TotalL, 2123
Trigger, 2451
Vlv2WayL, 1647
VlvAnL, 1808
VlvL, 1691
VlvMotL, 1754
VlvPosL, 1876
VlvS, 1728
XOr04, 2456
Obtaining the standard value
FbAnIn, 2468
FbAnOu, 2478
FbAnTot, 2488
FbDiIn, 2497
FbDiOu, 2507
Pcs7AITC, 2555
Pcs7AnIn, 2567
Pcs7DiIn, 2599
Pcs7DiIT, 2608
Pcs7HaAI, 2667
Pcs7HaAO, 2674
Online optimization of the PID controller parameters
FmTemp, 928
OpAnL
Area of application, 389
Associated values, 396
Block diagram, 403
Block icon, 410
Configurable reactions using the Feature parameter, 393
Configuration, 389
Considering bad quality of automatic commands or external values, 392
Error handling, 394
Forming the signal status for blocks, 391
Functions, 391
Gradient limit of the setpoint, 391
How it works, 389
I/Os, 397
Internal or external setpoint selection, 391
Messaging, 395
Object name, 389
Opening additional faceplates, 394
Operating modes, 390
Operator permissions, 393
Overview of error numbers, 395
Parameter view, 407
Preview, 409
Process messages, 396
Selecting a unit of measure, 392
Setpoint limitation, 391
SIMATIC BATCH functionality, 394
Simulating signals, 392
Specifying the display area for process and setpoint values as well as operations, 394
Standard view, 404
Startup characteristics, 389
Status word allocation, 389
Suppressing messages using the MsgLock parameter, 391
OpAnL trend view, 407
OpAnS
Area of application, 412
Block diagram, 420
Block icon, 425
Configurable reactions using the Feature parameter, 415
Configuration, 412
Considering bad quality of automatic commands or external values, 414
Error handling, 416
Forming the signal status for blocks, 414
Functions, 414
How it works, 412
I/Os, 417
Internal or external setpoint selection, 414
Messaging, 417
Object name, 412
Opening additional faceplates, 415
Operating modes, 413
Operator permissions, 415
Overview of error numbers, 416
Parameter view, 423
Preview, 424
Selecting a unit of measure, 414
SIMATIC BATCH functionality, 416

Simulating signals, 414
Specifying the display area for process and setpoint values as well as operations, 415
Standard view, 421
Startup characteristics, 412
Status word allocation, 412
OpDi01
Area of application, 426
Block diagram, 435
Block icon, 439
Changing labels on buttons and text, 428
Configurable reactions using the Feature parameter, 430
Configuration, 426
Considering bad quality of automatic commands or external values, 429
Error handling, 431
Forming the signal status for blocks, 429
Functions, 428
How it works, 426
I/Os, 432
Input parameter for feedback value, 428
Interlocks, 428
Internal or external digital value, 428
Messaging, 431
Object name, 426
Opening additional faceplates, 428
Operating modes, 427
Operator permissions, 429
Overview of error numbers, 431
Preview, 437
Standard view, 436
Startup characteristics, 426
Status word allocation, 426
OpDi03
Area of application, 441
Block diagram, 451
Block icon, 455
Changing labels on buttons and text, 444
Configurable reactions using the Feature parameter, 446
Configuration, 441
Considering bad quality of automatic commands or external values, 445
Error handling, 446
Forming the signal status for blocks, 444
Functions, 443
How it works, 441
I/Os, 448
Input parameter for feedback value, 444
Interlocks, 443
Internal or external digital value, 443
Messaging, 447
Object name, 441
Opening additional faceplates, 444
Operating modes, 442
Operator permissions, 445
Overview of error numbers, 447
Preview, 454
Resetting all output values, 444
Standard view, 452
Startup characteristics, 441
Status word allocation, 441
Opening additional faceplates, 234
ConPerMon, 856
CountOh, 2105
CountScL, 2083
DoseL, 1323
FmCont, 892
FmTemp, 933
Intlk02, 1953
Intlk04, 1968
Intlk08, 1982
Intlk16, 2000
ModPreCon, 986
MonAnL, 725
MonAnS, 755
MonDi08, 821
MonDiL, 778
MonDiS, 803
MotL, 1382
MotRevL, 1447
MotS, 1419
MotSpdCL, 1496
MotSpdL, 1563
MPC10x10, 1251
OpAnL, 394
OpAnS, 415
OpDi01, 428
OpDi03, 444
OpTrig, 472
PIDConL, 1034
PIDConR, 1117
PIDConS, 1079
PIDStepL, 1158
Ratio, 1200
SelA16In, 2364
SelAn16L, 2383
ShrdResL, 1607
ShrdResS, 1629
TotalL, 2133
Vlv2WayL, 1658
VlvAnL, 1817
VlvL, 1696

VlvMotL, 1762
VlvPosL, 1891
VlvS, 1732
Operating mode
Automatic for controllerblocks, 79
Automatic for motors, valves and dosers, 82
In progress, 70
Local mode, 86
Manual for controller blocks, 79
Manual for motors, valves and dosers, 82
On, 77
Out of service, 70, 77
Overview for status change, 90
Program mode, 85
Rcv_AnaVal, 2813
Rcv_DigVal, 2806
Snd_DigVal, 2802, 2810
Operating modes
AbsR, 2208
Add04, 2211
Add08, 2216
And04, 2400
And08, 2405
AV, 707
Average, 2221
CompAn02, 2320
ConPerMon, 845
CountOh, 2102
CountScL, 2079
DeadTime, 2228
Derivative, 2235
Div02, 2241
DoseL, 1310
Event, 2020
Event16Ts, 2062
EventNck, 2032
EventTs, 2045
FbAnIn, 2468
FbAnTot, 2488
FbDiIn, 2496
FbEnMe, 2529
FbSwtMMS, 2542
FirstIn, 2013
FlipFlop, 2412
FlowCorr, 2247
FmCont, 880
FmTemp, 920
GainSched, 961
Integral, 2254
KalFilt, 1282
Lag, 2263
Limit, 2326
MeanTime, 2269
ModPreCon, 975
MonAnL, 718
MonAnS, 752
MonDi08, 818
MonDiL, 773
MonDiS, 799
MotL, 1380
MotRevL, 1445
MotS, 1418
MotSpdCL, 1494
MotSpdL, 1561
MPC10x10, 1241
MSTIn, 2739
MSTOu, 2743
Mul04, 2276
Mul08, 2282
MuxAn03, 2331
MuxAn08, 2338
MuxMST, 2766
MuxST, 2771
Not01, 2428
OpAnL, 390
OpAnS, 413
OpDi01, 427
OpDi03, 442
OpStations, 459
OpTrig, 471
Or04, 2418
Or08, 2423
Pcs7Cnt2, 2639
Pcs7Cnt3, 2648
Pcs7HaAI, 2666
Pcs7HaAO, 2673
PIDConL, 1022
PIDConR, 1100
PIDConS, 1072
PIDStepL, 1146
Polygon, 2290
PowerXY, 2314
Psc7Cnt1, 2627
RateLim, 2345
Ratio, 1198
RedAn02, 2353
RedDi02, 2432
SelA02In, 2358
SelA16In, 2363, 2382
SelD02In, 2437
ShrdResL, 1606
ShrdResS, 1629
Smooth, 2298
SplRange, 1217

SqrRoot, 2304
STIn, 2731
STOu, 2735
StrctCom, 2442
StrctDeC, 2447
StrgToBy, 2702
StruScOu, 2727
Sub02, 2309
TimerP, 2174
TimeTrig, 2183
TotalL, 2132
Trigger, 2452
Vlv2WayL, 1651
VlvAnL, 1813
VlvL, 1694
VlvMotL, 1759
VlvPosL, 1882
VlvS, 1731
XOr04, 2457
Operating modes of AssetM, 2781
Operating modes of CntOhSc, 2158
Operating modes of RealToDw, 2748
Operating modes of STRep, 2776
Operating modes of the blocks
Overview, 76
Operating permissions
Intlk02, 1955
Intlk04, 1968
Intlk08, 1983
Intlk16, 2000
Operating principle
KalFilt, 1278
Operator control permissions
Blocks, 285
Event, 2021
EventNck, 2034
MonDi08, 820
OpStations, 460
OpTrig, 472
SelA16In, 2364
SelAn16L, 2383
Operator input area for process values and setpoints, 233
Operator permissions
ConPerMon, 855
CountOh, 2106
CountScL, 2082
DoseL, 1324
Event16Ts, 2063
EventTs, 2047
FmCont, 889
FmTemp, 931
KalFilt, 1285
ModPreCon, 985
MonAnL, 724
MonAnS, 754
MonDiL, 777
MonDiS, 802
MotL, 1382
MotRevL, 1447
MotS, 1419
MotSpdCL, 1497
MotSpdL, 1563
MPC10x10, 1249
OpAnL, 393
OpAnS, 415
OpDi01, 429
OpDi03, 445
PIDConL, 1031
PIDConR, 1114
PIDConS, 1077
PIDStepL, 1156, 1200
ShrdResL, 1607
TotalL, 2135
Vlv2WayL, 1658
VlvAnL, 1827
VlvL, 1696
VlvMotL, 1762
VlvPosL, 1891
VlvS, 1732
OpStations
Area of application, 457
Block diagram, 464
Block icon, 469
Configuration, 457
Error handling, 461
Functions, 460
How it works, 457
I/Os, 462
Messaging, 461
Object name, 457
Operating modes, 459
Operator control permissions, 460
Startup characteristics, 458
Status word allocation, 459
OpTrig
Area of application, 470
Block diagram, 477
Block icon, 480
Configurable reactions using the Feature parameter, 473
Configuration, 470
Considering bad quality of automatic commands or external values, 472

Error handling, 473
Forming the signal status for blocks, 472
Functions, 471
How it works, 470
I/Os, 474
Input parameter for feedback value, 472
Issuing trigger signal internally or externally, 472
Messaging, 474
Object name, 470
Opening additional faceplates, 472
Operating modes, 471
Operator control permissions, 472
Preview, 479
Simulating signals, 473
Standard view, 478
Startup characteristics, 470
Status word allocation, 470
Or04
Area of application, 2417
Block diagram, 2421
Configuration, 2417
Error handling, 2419
Functions, 2419
How it works, 2417
I/Os, 2420
Messaging, 2420
Object name, 2417
Operating modes, 2418
Startup characteristics, 2418
Status word allocation, 2418
Or08
Area of application, 2422
Block diagram, 2426
Configuration, 2422
Error handling, 2424
Functions, 2423
How it works, 2422
I/Os, 2425
Messaging, 2424
Object name, 2422
Operating modes, 2423
Startup characteristics, 2422
Status word allocation, 2423
Out of service
Operating mode description, 77
Output of invalid value if raw value is invalid
FbDiIn, 2497
Pcs7AnIn, 2569
Pcs7DiCnt, 2590
Pcs7DiIn, 2599
Pcs7DiIT, 2608
Output signal as a pulse signal or static signal
DoseL, 1314
MotL, 1387
MotRevL, 1454
MotS, 1422
MotSpdCL, 1507
MotSpdL, 1571
Vlv2WayL, 1653
VlvL, 1703
VlvMotL, 1771
Output signal as pulse signal or static signal
VlvPosL, 1903
Output substitute value if raw value is invalid
FbAnIn, 2469
FbAnTot, 2488
FbDiIn, 2497
Pcs7AITC, 2555
Pcs7AnIn, 2569
Pcs7DiCnt, 2590
Pcs7DiIn, 2599
Pcs7DiIT, 2608
Outputting a signal for start readiness
DoseL, 1321
FmCont, 884
FmTemp, 924
MotL, 1384, 1422, 1502
MotRevL, 1450
MotSpdL, 1566
PIDConL, 1025
PIDConR, 1105
PIDConS, 1074
PIDStepL, 1150
Vlv2WayL, 1656
VlvAnL, 1818
VlvL, 1698
VlvMotL, 1767
VlvPosL, 1898
Outputting an invalid value if analog value is invalid
FbAnIn, 2469
FbAnTot, 2489
Overdosing/underdosing
DoseL, 1317
Override control
ConPerMon, 852
Override control (override), 1017, 1094, 1141
Overshoot, 842, 848, 857, 869, 2875
Overview of error numbers, 128
AV, 709
Average, 2223
ConPerMon, 857
CountOh, 2107
CountScL, 2084

DeadTime, 2229
Derivative, 2237
Div02, 2242
DoseL, 1327
Event, 2022
EventNck, 2035
EventTs, 2048, 2065
FmCont, 893
FmTemp, 934
Integral, 2257
Intlk02, 1957
Intlk04, 1970
Intlk08, 1985
Intlk16, 2002
KalFilt, 1286
Lag, 2265
MeanTime, 2272
ModPreCon, 989
MonAnL, 726, 779
MonAnS, 756
MonDi08, 822
MonDiS, 803
MotL, 1391
MotRevL, 1458
MotS, 1424
MotSpdCL, 1512
MotSpdL, 1575
MPC10x10, 1254
OpAnL, 395
OpAnS, 416
OpDi01, 431
OpDi03, 447
PIDConL, 1035
PIDConR, 1117
PIDConS, 1080
PIDStepL, 1159
Polygon, 2291
RateLim, 2348
Ratio, 1202
SelA16In, 2367
SelAn16L, 2385
ShrdResL, 1613
ShrdResS, 1632
Smooth, 2300
SplRange, 1221
TimerP, 2177
TimeTrig, 2191
TotalL, 2138
Vlv2WayL, 1661
VlvAnL, 1832
VlvL, 1704
VlvMotL, 1775
VlvPosL, 1908
VlvS, 1737

## P

P action, 81, 224
P component, 222, 887
P controller, 970, 1235
P step change, 184
PA/FF devices
  Variant identifier, 2698
PA/FF devices variant identifier, 2698
PA/FF_MODE
  Settings, 2698
Parameter view
  DoseL, 1356
  FM controller, 318
  GainSched, 968
  ModPreCon, 1010
  Motor, 320
  MPC10x10, 1269, 1270
  PID controller, 315
  Ratio, 1211
  ShrdResL, 1624
  Vlv2WayL, 1681
PCS 7 measuring point browser, 383
PCS 7 multiproject, 2818
PCS 7 PID tuner, 2825
Pcs7AITC
  Area of application, 2551
  Channel error, 2558
  Checking the raw value, 2554
  Configurable reactions using the Feature parameter, 2556
  Configuration, 2551
  Error handling, 2558
  Flutter suppression, 2556
  Functions, 2554
  Higher-level error / invalid measuring range, 2559
  Holding the last value if raw value is invalid, 2555
  How it works, 2551
  Object name, 2551
  Obtaining the standard value, 2555
  Output substitute value if raw value is invalid, 2555
  Quality code, 2554
  Redundancy, 2554
  Scalable measuring range (only ET 200SP), 2557
  Signal status for PCS 7 channel blocks, 2556
  Sign-of-life monitoring, 2557
  Simulating signals, 2556
  Startup characteristics, 2553

Status word allocation, 2554
Value application delay, 2556
Pcs7AnIn
Area of application, 2564
Block diagram, 2576
Channel error, 2572
Configurable reactions using the Feature parameter, 2570
Configuration, 2564
Error handling, 2571
Functions, 2567
Higher-level error / invalid measuring range, 2572
Hold last value, 2568
How it works, 2564
I/Os, 2572
Messaging, 2572
Modes, 2566
Object name, 2564
Obtaining the standard value, 2567
Output of invalid value if raw value is invalid, 2569
Output substitute value if raw value is invalid, 2569
Quality code, 2566
Raw value check, 2567
Redundancy, 2566
Signal status for PCS 7 channel blocks, 2569
Simulating signals, 2570
Startup characteristics, 2566
Status word allocation, 2566
PCS7AnIn
Value application delay, 2569
Pcs7AnOu
Area of application, 2577
Block diagram, 2587
Channel error, 2583
Configurable reactions using the Feature parameter, 2582
Configuration, 2577
Error handling, 2582
Flutter suppression, 2581
Forming an I/O value, 2580
Forming the signal status for PCS 7 channel blocks, 2581
Functions, 2580
Higher-level error / invalid measuring range, 2583
How it works, 2577
I/Os, 2584
Keep last valid raw value, 2581
Limiting the peripheral value, 2581
Limiting the process value, 2581
Messaging, 2583
Modes, 2580
Object name, 2577
Quality code, 2579
Redundancy, 2579
Simulating signals, 2581
Startup characteristics, 2579
Status word allocation, 2579
Pcs7Cnt2, 2636
Area of application, 2636
Block diagram, 2645
Configurable reactions using the Feature parameter, 2640
Configuration, 2638
Description, 2636
Error handling, 2641
Flutter suppression, 2640
Functions, 2639
How it works, 2636
I/Os, 2642
Messaging, 2642
Operating modes, 2639
Signal status, 2641
Startup characteristics, 2639
Status word allocation, 2639
Pcs7Cnt3
Area of application, 2646
Block diagram, 2654
Configurable reactions using the Feature parameter, 2649
Configuration, 2647
Description of, 2646
Error handling, 2649
Functions, 2648
How it works, 2646
I/Os, 2650
Messaging, 2650
Operating modes, 2648
Signal status, 2649
Startup characteristics, 2647
Status word allocation, 2647
Pcs7DiCnt
Area of application, 2588
Channel error, 2591
Configurable reactions using the Feature parameter, 2590
Configuration, 2588
Error handling, 2591
Functions, 2589
Higher-level error / invalid measuring range, 2591
Holding the last value if raw value is invalid, 2590
How it works, 2588
Messaging, 2591

Object name, 2588
Output of invalid value if raw value is invalid, 2590
Output substitute value if raw value is invalid, 2590
Process values in Count and frequency mode, 2589
Signal status for PCS 7 channel blocks, 2590
Simulating signals, 2590
Startup characteristics, 2589
Status word allocation, 2589
Pcs7DiIn
Area of application, 2596
Block diagram, 2604
Channel error, 2600
Configurable reactions using the Feature parameter, 2599
Configuration, 2596
Error handling, 2600
Flutter suppression, 2599
Functions, 2598
Higher-level error / invalid measuring range, 2601
Hold last value, 2599
How it works, 2596
I/Os, 2601
Messaging, 2601
Modes, 2598
Object name, 2596
Obtaining the standard value, 2599
Output of invalid value if raw value is invalid, 2599
Output substitute value if raw value is invalid, 2599
Quality code, 2597
Redundancy, 2597
Signal status for PCS 7 channel blocks, 2599
Simulating signals, 2599
Startup characteristics, 2597
Status word allocation, 2598
Pcs7DiIT
Area of application, 2605
Block diagram, 2614
Configurable reactions using the Feature parameter, 2609
Configuration, 2605
Error handling, 2610
Functions, 2607
Hold last value, 2608
How it works, 2605
I/Os, 2611
Messaging, 2610
Modes, 2607
Object name, 2605
Obtaining the standard value, 2608
Output of invalid value if raw value is invalid, 2608
Output substitute value if raw value is invalid, 2608
Quality code, 2606
Redundancy, 2606
Signal status for PCS 7 channel blocks, 2608
Simulating signals, 2608
Startup characteristics, 2606
Status word allocation, 2607
Time stamp, 2608
Pcs7DiOu
Area of application, 2616
Block diagram, 2624
Configurable reactions using the Feature parameter, 2619
Configuration, 2616
Error handling, 2620
Flutter suppression, 2619
Forming a peripheral value, 2619
Functions, 2619
Higher-level error / invalid measuring range, 2620
How it works, 2616
I/Os, 2621
Keep last valid raw value, 2619
Messaging, 2621
Modes, 2618
Object name, 2616
Quality code, 2618
Redundancy, 2618
Simulating signals, 2619
Startup characteristics, 2617
Status word allocation, 2618
Pcs7HaAI
Area of application, 2665
Block diagram, 2671
Configuration, 2665
Error handling, 2667
Functions, 2666
How it works, 2665
I/Os, 2668
Invalid value, 2667
Messaging, 2668
Object name, 2665
Obtaining the standard value, 2667
Operating modes, 2666
Startup characteristics, 2666
Status word allocation, 2666
Pcs7HaAO
Area of application, 2672

Block diagram, 2678
Configuration, 2672
Error handling, 2674
Functions, 2673
How it works, 2672
I/Os, 2675
Invalid value, 2674
Messaging, 2675
Object name, 2672
Obtaining the standard value, 2674
Operating modes, 2673
Startup characteristics, 2673
Status word allocation, 2673
Periodic output of a trigger
TimeTrig, 2184
Physical standardization of setpoint, manipulated variable and process value
FmCont, 886
FmTemp, 926
PIDConL, 1027
PIDConR, 1108
PIDStepL, 1151
PI controller, 2830
PID algorithm
FmCont, 887
FmTemp, 927
PIDConL, 1027
PIDConR, 1109
PIDConS, 1075
PIDStepL, 1152
PID controller, 85, 841, 959, 970, 979, 1017, 1069, 1094, 1216, 1245, 2822, 2825, 2827, 2833, 2875
PID controller with gain scheduler
ConPerMon, 852
PID controllers, 975, 977, 1241, 1243, 2840, 2843, 2848
PIDCoefR
Area of application, 2794
Object name, 2794
PIDConL
Actuator active information, 1025
Anti-windup, 1028
Area of application, 1017
Associated values, 1038
Block diagram, 1058
Block icon, 269
Bypass function, 1026
Configurable reactions using the Feature parameter, 1030
Configurable reactions using the Feature2 parameter, 1031
Configuration, 1017
Control zone, 1029
Error handling, 1034
Error signal generation and dead band, 1026
External/internal setpoint specification, 1025
Feedforwarding and limiting disturbance variables, 1028
Forming the signal status for blocks, 1029
Functions, 1024
Generating instance-specific messages, 1033
Generation of manipulated variables, 1024
Gradient limit of the setpoint, 1025
Group error, 1025
How it works, 1017
I/Os, 1039
Instance-specific messages, 1037
Interlocks, 1033, 1116
Inverting control direction, 1027
Limit monitoring of error signal, 1027
Limit monitoring of position feedback, 1025
Limit monitoring of the process value, 1026
Messaging, 1036
Neutral position, 1024
Object name, 1017
Opening additional faceplates, 1034
Operating modes, 1022
Operator permissions, 1031
Outputting a signal for start readiness, 1025
Overview of error numbers, 1035
Physical standardization of setpoint, manipulated variable and process value, 1027
PID algorithm, 1027
Preview, 1066
Process control fault, 1036
Process messages, 1037, 1911
Release for maintenance, 1033
Selecting a unit of measure, 1027
Setpoint limiting for external setpoints, 1025
Setpoint ramp, 1025
SIMATIC BATCH functionality, 1034
Simulating signals, 1026
Specifying the display area for process and setpoint values as well as operations, 1034
Startup characteristics, 1019
Status word allocation, 1019
Structure segmentation at controllers, 1028
Suppressing messages using the MsgLock parameter, 1033
Time-stamped messages, 1034
Tracking and limiting a manipulated variable, 1024
Tracking the setpoint, 1025

PIDConR
Actuator active information, 1105
Anti-windup, 1112
Area of application, 1094
Associated values, 1120
Block diagram, 1140
Bumpless switchover between internal and external setpoint with PID initialization, 1107
Bumpless switchover from external to internal setpoint, 1107
Bypass function, 1107
Configurable reactions using the Feature parameter, 1113
Configuration, 1095
Displaying additional information relating to the manipulated variable on the output, 1105
Error handling, 1117
Error signal generation and dead band, 1108
External/internal setpoint specification, 1106
Feedforwarding and limiting disturbance variables, 1112
Forming the signal status for blocks, 1112
Functions, 1103
Generating instance-specific messages, 1116
Generation of manipulated variables, 1103
Gradient limit of the setpoint, 1107
Group error, 1105
How it works, 1094
I/Os, 1121, 1163, 1164, 1165, 1166, 1167, 1168, 1169, 1170, 1171, 1172, 1173
Instance-specific messages, 1120
Inverting control direction, 1108
Limit monitoring of error signal, 1108
Limit monitoring of position feedback, 1106
Limit monitoring of the process value, 1107
Messaging, 1118
Neutral position, 1105
Object name, 1094
Opening additional faceplates, 1117
Operating modes, 1100
Operator permissions, 1114
Outputting a signal for start readiness, 1105
Overview of error numbers, 1117
Physical standardization of setpoint, manipulated variable and process value, 1108
PID algorithm, 1109
Preview, 1066
Process control fault, 1119
Process messages, 1119
Release for maintenance, 1116
Selecting a unit of measure, 1109
Setpoint limiting for external setpoints, 1107
Setpoint ramp, 1107
SIMATIC BATCH functionality, 1117
Simulating signals, 1107
Specifying the display area for process and setpoint values as well as operations, 1117
Startup characteristics, 1096
Status word allocation, 1097
Suppressing messages using the MsgLock parameter, 1116
Time-stamped messages, 1117
Tracking and limiting a manipulated variable, 1105
Tracking the setpoint, 1107
Use output point for the manipulated variable calculation, 1111
PIDConS
Actuator active information, 1074
Anti-windup, 1076
Area of application, 1069
Associated values, 1082
Block diagram, 1091
Block icon, 269
Configurable reactions using the Feature parameter, 1077
Configuration, 1069
Error handling, 1079
Error signal generation and dead band, 1075
External/internal setpoint specification, 1075
Forming the signal status for blocks, 1076
Functions, 1073
Generating instance-specific messages, 1079
Generation of manipulated variables, 1074
Group error, 1074
How it works, 1069
I/Os, 1082
Instance-specific messages, 1081
Limit monitoring of the process value, 1075
Messaging, 1080
Neutral position, 1074
Object name, 1069
Opening additional faceplates, 1079
Operating modes, 1072
Operator permissions, 1077
Outputting a signal for start readiness, 1074
Overview of error numbers, 1080
PID algorithm, 1075
Preview, 1066
Process control fault, 1081
Process messages, 1081
Release for maintenance, 1078
Selecting a unit of measure, 1075
Setpoint limitation, 1075

SIMATIC BATCH functionality, 1079
Simulating signals, 1075
Specifying the display area for process and setpoint values as well as operations, 1079
Startup characteristics, 1070
Status word allocation, 1070
Suppressing messages using the MsgLock parameter, 1079
Tracking and limiting a manipulated variable, 1074
Tracking the setpoint, 1075
PIDKernR
Area of application, 2800
Object name, 2800
PIDStepL
Actuator active information, 1150
Anti-windup, 1153
Area of application, 1141
Associated values, 1162
Block diagram, 1179
Button labels, 1158
Bypass function, 1151
Configurable reactions using the Feature parameter, 1155
Configurable reactions using the Feature2 parameter, 1156
Configuration, 1142
Error handling, 1159
Error signal generation and dead band, 1151
External/internal setpoint specification, 1150
Feedforwarding and limiting disturbance variables, 1153
Forming the signal status for blocks, 1154
Functions, 1147
Generating instance-specific messages, 1158
Generation of actuating signal without position feedback, 1148
Generation of manipulated variables, 1148
Gradient limit of the setpoint, 1150
Group error, 1150
How it works, 1141
Instance-specific messages, 1162
Inverting control direction, 1151
Limit monitoring of error signal, 1151
Limit monitoring of position feedback, 1150
Limit monitoring of the process value, 1151
Messaging, 1160
Neutral position, 1149
Object name, 1141
Opening additional faceplates, 1158
Operating modes, 1146
Operator permissions, 1156, 1200
Outputting a signal for start readiness, 1150
Overview of error numbers, 1159
Physical standardization of setpoint, manipulated variable and process value, 1151
PID algorithm, 1152
Preview, 1193
Process control fault, 1161
Process messages, 1161
Release for maintenance, 1158
Selecting a unit of measure, 1152
Setpoint limiting for external setpoints, 1150
Setpoint ramp, 1150
SIMATIC BATCH functionality, 1158
Simulating signals, 1151
Specifying the display area for process and setpoint values as well as operations, 1158
Startup characteristics, 1142
Status word allocation, 1143
Structure segmentation at controllers, 1153
Suppressing messages using the MsgLock parameter, 1158
Time-stamped messages, 1159
Tracking and limiting a manipulated variable, 1149
Tracking the setpoint, 1151
Polygon
Area of application, 2287
Block diagram, 2296
Configuration, 2289
Error handling, 2291
Forming the signal status for blocks, 2290
Functions, 2290
How it works, 2288
I/Os, 2293
Messaging, 2292
Object name, 2287
Operating modes, 2290
Overview of error numbers, 2291
Startup characteristics, 2289
Status word allocation, 2289
Positive edge, 59
Post dosing
DoseL, 1318
PowerXY
Area of application, 2313
Block diagram, 2317
Configuration, 2314
Error handling, 2315
Forming the signal status for blocks, 2315
Functions, 2315
How it works, 2313
I/Os, 2316

Messaging, 2316
Operating modes, 2314
Startup characteristics, 2314
Status word allocation, 2314
Prediction horizon, 2883
Predictive controller, 977, 1244
Predictive controller algorithm
ModPreCon, 979
MPC10x10, 1245
Preview
ModPreCon, 1011
MPC10x10, 1271
ShrdResL, 1622
Preview of CntOhSc, 2169
Process control fault, 894, 935, 1036, 1080, 1118, 1160
DoseL, 1329
FmCont, 894
FmTemp, 935
MonAnL, 727
MonAnS, 756
MonDiL, 780
MonDiS, 804
MotL, 1392
MotRevL, 1461
MotS, 1425
MotSpdCL, 1516
MotSpdL, 1577
PIDConL, 1036
PIDConR, 1119
PIDConS, 1081
PIDStepL, 1161
Vlv2WayL, 1663
VlvAnL, 1833
VlvL, 1706
VlvMotL, 1777
VlvS, 1738
Process control messages, 283
Process dead time, 851
Process messages, 894, 935, 1036, 1080, 1118, 1160
AV, 710
ConPerMon, 858
CountOh, 2108
CountScL, 2085
DoseL, 1330
Event, 2023
EventNck, 2036
EventTs, 2049
FmCont, 895
FmTemp, 936
MonAnL, 727
MonAnS, 757
MonDi08, 823
MonDiL, 780
MonDiS, 804
OpAnL, 396
PIDConL, 1037, 1911
PIDConR, 1119
PIDConS, 1081
PIDStepL, 1161
TotalL, 2139
VlvAnL, 1834
Process messages (MsgEvId1)
Event16Ts, 2066
Process messages (MsgEvId2)
Event16Ts, 2067
Process simulation, 2871, 2875, 2876, 2877
Process value with separate scale range, 173
Process values in Count and frequency mode
Pcs7DiCnt, 2589
Program mode
Description, 85
Proportional gain, 183
Providing limits at the output, 1026, 1108, 1151, 1319, 1499, 1830, 1893
Psc7AnIn
Flutter suppression, 2569
Psc7Cnt1, 2625
Area of application, 2625
Block diagram, 2635
Configurable reactions using the Feature parameter, 2628
Configuration, 2626
Description, 2625
Error handling, 2629
Flutter suppression, 2628
Forming the signal status for blocks, 2634
Functions, 2627
How it works, 2625
I/Os, 2630
Messaging, 2630
Operating modes, 2627
Signal status, 2628
Startup characteristics, 2626
Status word allocation, 2626
Psc7Cnt3
Flutter suppression, 2648
Psc7DiIT
Flutter suppression, 2608
Pulse controller, 876, 882, 916, 922
Pulse signal with configurable pulse length, 57

## Q

Quality code
Pcs7AITC, 2554
Pcs7AnIn, 2566
Pcs7AnOu, 2579
Pcs7DiIn, 2597
Pcs7DiIT, 2606
Pcs7DiOu, 2618

## R

R64ToReal
Area of application, 2795
Object name, 2795
Ramp function, 335
Ramp view, 131, 133, 334
Rapid stop, 115
Description, 115
MotL, 1384
MotRevL, 1449
MotSpdCL, 1502
MotSpdL, 1565
VlvMotL, 1764
VlvPosL, 1895
RateLim
Area of application, 2344
Block diagram, 2351
Configurable reactions using the Feature parameter, 2347
Configuration, 2344
Error handling, 2348
Functions, 2345
How it works, 2344
I/Os, 2349
Limitation of the slope of an analog signal, 2346
Messaging, 2348
Object name, 2344
Operating modes, 2345
Overview of error numbers, 2348
Startup characteristics, 2345
Status word allocation, 2345
Stopping the caculation of the ramp output value, 2347
Switching the limiting function on or off, 2347
Ratio
Area of application, 1196
Block diagram, 1207
Block icon, 1213
Bumpless switchover from external to internal ratio, 1199
Configurable reactions using the Feature parameter, 1201
Configuration, 1197
Display and operator input area for process values and setpoints, 1199
Error handling, 1201
Forming and outputting signal status for blocks, 1200
Functions, 1198
How it works, 1196
I/Os, 1202
Internal or external ratio, 1199
Limiting the output value, 1199
Limiting the ratio, 1199
Messaging, 1202
Object name, 1196
Opening additional faceplates, 1200
Operating modes, 1198
Overview of error numbers, 1202
Preview, 1212
Selecting a unit of measure, 1199
Simulating signals, 1199
Standard view, 1208
Startup characteristics, 1197
Status word allocation, 1197
Ratio block, 2836
Ratio control, 876, 916, 970, 1017, 1094, 1141, 2836
ConPerMon, 853
Raw value check
Pcs7AnIn, 2567
Rcv_AnaVal
Area of application, 2812
Configuration, 2813
Error handling, 2814
Functions, 2813
How it works, 2812
Messaging, 2814
Object name, 2812
Operating mode, 2813
Startup characteristics, 2813
Rcv_DigVal
Area of application, 2805
Configuration, 2806
Error handling, 2807
Functions, 2807
How it works, 2805
Messaging, 2807
Object name, 2805
Operating mode, 2806

Startup characteristics, 2806
Time characteristics, 2806
Read back the last counted value
CountOh, 2104
CountScL, 2080
Readback of the most recently calculated sum
TotalL, 2136
Readiness signal
ShrdResL, 1608
ShrdResS, 1629
RealToR64
Area of application, 2796
Object name, 2796
Recording the first signal
Activate, 161
Interlock block, 58
RedAn02
Area of application, 2352
Block diagram, 2356
Configuration, 2352
Error handling, 2354
Forming the signal status for blocks, 2353
Functions, 2353
How it works, 2352
I/Os, 2354
Messaging, 2354
Object name, 2352
Operating modes, 2353
Startup characteristics, 2352
Status word allocation, 2352
RedDi02
Area of application, 2431
Block diagram, 2435
Configuration, 2431
Error handling, 2433
Forming the signal status for blocks, 2432
Functions, 2432
How it works, 2431
I/Os, 2433
Messaging, 2433
Object name, 2431
Operating modes, 2432
Startup characteristics, 2431
Status word allocation, 2431
Redundancy
FbAnIn, 2467
FbAnOu, 2477
Pcs7AITC, 2554
Pcs7AnIn, 2566
Pcs7AnOu, 2579
Pcs7DiIn, 2597
Pcs7DiIT, 2606
Pcs7DiOu, 2618
Release for maintenance, 70
AV, 708
CountOh, 2107
CountScL, 2083
DoseL, 1322
Event, 2021
Event16Ts, 2063
EventNck, 2034
EventTs, 2046
FmCont, 891
FmTemp, 933
ModPreCon, 986
MonAnL, 723
MonAnS, 753
MonDi08, 820
MonDiL, 777
MonDiS, 802
MotL, 1386, 1423
MotRevL, 1453
MotSpdCL, 1506
MotSpdL, 1570
MPC10x10, 1250
PIDConL, 1033
PIDConR, 1116
PIDConS, 1078
PIDStepL, 1158
TotalL, 2137
Vlv2WayL, 1657
VlvL, 1700
VlvMotL, 1770
VlvPosL, 1902
VlvS, 1735
Request for maintenance release
Issuing a request for maintenance (MS → APL), 68
Reset
Block, 47
Monitoring errors, 48
Reset counter to zero
CountOh, 2105
CountScL, 2080
Reset of motor control after valve end position has been reached
VlvMotL, 1765
Reset values
Lag, 2264
Resetting all output values
OpDi03, 444
Resetting the block in case of interlocks
VlvL, 1697
VlvMotL, 1766

VlvPosL, 1897
VlvS, 1733
Resetting the block in case of interlocks or errors
DoseL, 1321
MotL, 1384
MotRevL, 1449
MotS, 1421
MotSpdCL, 1502
MotSpdL, 1565
Vlv2WayL, 1656
VlvAnL, 1817
Resetting the dosing quantity
DoseL, 1318
Resetting the Out and TimeRemaining output parameters
TimerP, 2176
Restart low pass filter
Smooth, 2299

## S

S7_unit, 241
Scalable measuring range (only ET 200SP)
Pcs7AITC, 2557
Sealing the valve
VlvMotL, 1765
VlvPosL, 1896
SelA02In
Area of application, 2357
Block diagram, 2361
Configuration, 2357
Error handling, 2359
Functions, 2358
How it works, 2357
I/Os, 2360
Messaging, 2360
Object name, 2357
Operating modes, 2358
Select input parameter, 2359
Startup characteristics, 2357
Status word allocation, 2357
SelA16In
Area of application, 2362
Block diagram, 2372
Block icon, 2377
Configurable reactions using the Feature parameter, 2366
Configuration, 2362
Error handling, 2367
Forming the signal status for blocks, 2365
Functions, 2364
How it works, 2362
I/Os, 2368
Messaging, 2368
Object name, 2362
Opening additional faceplates, 2364
Operating modes, 2363, 2382
Operator control permissions, 2364
Overview of error numbers, 2367
Preview, 2376
Selecting a unit of measure, 2366
Standard view, 2374
Startup characteristics, 2362
Status word allocation, 2363
SelAn16L
Area of application, 2380
Block diagram, 2392
Block icon, 2397
Configurable reactions using the Feature parameter, 2384
Configuration, 2380
Error handling, 2385
Forming the signal status for blocks, 2383
Functions, 2382
How it works, 2380
I/Os, 2386
Messaging, 2386
Object name, 2380
Opening additional faceplates, 2383
Operator control permissions, 2383
Overview of error numbers, 2385
Selecting a unit of measure, 2384
Standard view, 2394
Startup characteristics, 2381
Status word allocation, 2381
SelD02In
Area of application, 2436
Block diagram, 2440
Configuration, 2436
Display of selected value, 2437
Error handling, 2438
Functions, 2437
How it works, 2436
I/Os, 2439, 2454
Messaging, 2438
Object name, 2436
Operating modes, 2437
Select input parameter, 2437
Startup characteristics, 2436
Status word allocation, 2436
Select input parameter
SelA02In, 2359
SelD02In, 2437

Select unit of measure
VlvPosL, 1903
Selecting a unit of measure
AV, 708
ConPerMon, 854
CountScL, 2081
DoseL, 1323
FmCont, 886
FmTemp, 927
GainSched, 962
ModPreCon, 978
MonAnL, 723
MonAnS, 754
MotL, 1387
MotRevL, 1454
MotSpdCL, 1506
MotSpdL, 1571
MPC10x10, 1244
OpAnL, 392
OpAnS, 414
PIDConL, 1027
PIDConR, 1109
PIDConS, 1075
PIDStepL, 1152
Ratio, 1199
SelA16In, 2366
SelAn16L, 2384
TotalL, 2133
Vlv2WayL, 1658
VlvAnL, 1823
VlvL, 1700
VlvMotL, 1770
Selecting signals for processing
MuxST, 2771
Selection of output signal
MuxAn03, 2331
MuxAn08, 2339
SelST16
Area of application, 2797
Object name, 2797
Setpoint adaption in motor stop and starting
Setpoint adaption in motor stop and starting, 1499
Setpoint difference, 102
Setpoint filters
ModPreCon, 977
MPC10x10, 1244
Setpoint input
External, 136
Internal, 136
Setpoint limitation
DoseL, 1319
MotSpdCL, 1499
OpAnL, 391
PIDConS, 1075
Setpoint limiting for external setpoints
FmCont, 885
FmTemp, 925
PIDConL, 1025
PIDConR, 1107
PIDStepL, 1150
Setpoint ramp, 876, 916
MotSpdCL, 1500
PIDConL, 1025
PIDConR, 1107
PIDStepL, 1150
Using, 131
Setpoint specification
With separate display area and custom unit, 182
Setpoint tracking in manual mode
ModPreCon, 977
MPC10x10, 1244
Setpoint view
ConPerMon, 872
DoseL, 1358
Setpoint view (Feature bit 15 set)
DoseL, 1360
Setting a mean value constant
MeanTime, 2270
Setting the count value to the default setting
CountOh, 2105
CountScL, 2081
Setting the setpoint internally
ModPreCon, 977
MPC10x10, 1243
Setting the summing/integrating value to the default
TotalL, 2137
Setting the time
TimerP, 2176
Setting warning times, 56
ShLeInt64
Area of application, 2798
Object name, 2798
ShrdResL
Allocate/enable channel, 1610
Area of application, 1603
Block diagram, 1619
Block icon, 1625
Channel management, 1609
Channel prioritization, 1611
Configuration, 1604
Enable/disable channel, 1611
Error handling, 1613
Functions, 1607

General preview, 1622
How it works, 1603
I/Os, 1614
Object name, 1603
Opening additional faceplates, 1607
Operating modes, 1606
Operator permissions, 1607
Overview of error numbers, 1613
Parameter view, 1624
Readiness signal, 1608
SIMATIC BATCH functionality, 1611
Startup characteristics, 1604
Status word allocation, 1605
ShrdResS
Allocate/enable channel, 1630
Block diagram, 1641
Block icon, 1645
Cascading, 1631
Channel management, 1630
Channel prioritization, 1631
Configuration, 1627
Enable/disable channel, 1631
Error handling, 1632
Functions, 1629
How it works, 1626
I/Os, 1633
Messaging, 1632
Object name, 1626
Opening additional faceplates, 1629
Operating modes, 1629
Overview of error numbers, 1632
Readiness signal, 1629
SIMATIC BATCH functionality, 1631
Startup characteristics, 1627
Status word allocation, 1628
ShrdResS ShrdResS
Area of application, 1626
ShRiInt64
Area of application, 2799
Object name, 2799
Signal status
AssetM, 2781
Pcs7Cnt3, 2649
PcsCnt1, 2628
PcsCnt2, 2641
Signal status as associated value of a message
Event16Ts, 2063
EventTs, 2046
Signal status for Fb channel blocks
FbAnIn, 2469
FbAnIOu, 2507
FbAnOu, 2478
FbAnTot, 2489
FbDiIn, 2497
Signal status for PCS 7 channel blocks
Pcs7AITC, 2556
Pcs7AnIn, 2569
Pcs7AnOu, 2581
Pcs7DiCnt, 2590
Pcs7DiIn, 2599
Pcs7DiIT, 2608
Signal status of the block
Description, 116
Overview of the values, 116
Sign-of-life monitoring
Pcs7AITC, 2557
SIMATIC BATCH, 73, 960
SIMATIC BATCH functionality, 73
ConPerMon, 856
CountOh, 2107
CountScL, 2083
DoseL, 1326
FmCont, 892
FmTemp, 933
ModPreCon, 986
MonAnL, 725
MonAnS, 755
MonDi08, 821
MonDiL, 778
MonDiS, 803
MotL, 1390
MotRevL, 1457
MotS, 1423
MotSpdCL, 1510
MotSpdL, 1574
MPC10x10, 1251
OpAnL, 394
OpAnS, 416
PIDConL, 1034
PIDConR, 1117
PIDConS, 1079
PIDStepL, 1158
ShrdResL, 1611
ShrdResS, 1631
TotalL, 2137
Vlv2WayL, 1660
VlvAnL, 1818
VlvL, 1703
VlvMotL, 1773
VlvPosL, 1906
VlvS, 1736
Simulating signals
AV, 708
DoseL, 1320

FbAnIn, 2469, 2498
FbAnOu, 2479
FbAnTot, 2489
FbDiOu, 2508
FmCont, 885
FmTemp, 925
General description, 64
ModPreCon, 978
MonAnL, 723
MonAnS, 753
MonDi08, 820
MonDiL, 777
MonDiS, 802
MotL, 1387
MotRevL, 1454
MotS, 1422
MotSpdCL, 1506
MotSpdL, 1570
MPC10x10, 1244
OpAnL, 392
OpAnS, 414
OpTrig, 473
Pcs7AITC, 2556
Pcs7AnIn, 2570
Pcs7AnOu, 2581
Pcs7DiCnt, 2590
Pcs7DiIn, 2599
Pcs7DiIT, 2608
Pcs7DiOu, 2619
PIDConL, 1026
PIDConR, 1107
PIDConS, 1075
PIDStepL, 1151
Ratio, 1199
TotalL, 2134
Vlv2WayL, 1655
VlvAnL, 1818
VlvL, 1700
VlvMotL, 1770
VlvPosL, 1903
VlvS, 1735
Smith predictor
ConPerMon, 852
Smith predictor closed-loop control, 1017, 1094, 1141
Smith predictors, 2830, 2877
Smooth
Activate and deactivate maverick detection, 2299
Area of application, 2297
Block diagram, 2302
Configuration, 2297
Error handling, 2299
Forming the signal status for blocks, 2299
Functions, 2298
How it works, 2297
I/Os, 2301
Messaging, 2300
Object name, 2297
Operating modes, 2298
Overview of error numbers, 2300
Restart low pass filter, 2299
Startup characteristics, 2298
Status word allocation, 2298
Smoothing PV
MonAnL, 722
Snd_AnaVal
Area of application, 2809
Error handling, 2810
Functions, 2810
How it works, 2809
Messaging, 2811
Object name, 2809
Snd_DigVal
Area of application, 2801
Configuration, 2802, 2810
Error handling, 2803
Functions, 2802
How it works, 2801
Messaging, 2803
Object name, 2801
Operating mode, 2802, 2810
Startup characteristics, 2802, 2810
Time characteristics, 2802, 2810, 2813
Specify warning times for control functions
MotL, 1387
MotRevL, 1453
MotSpdCL, 1506
MotSpdL, 1570
Vlv2WayL, 1654
VlvL, 1700
VlvMotL, 1770
VlvPosL, 1902
Specifying how TimerP works
TimerP, 2175
Specifying the display and operator input area for process values and setpoints
MotSpdCL, 1506
Specifying the display area for process and setpoint values as well as operations
FmCont, 891
FmTemp, 933
ModPreCon, 986
MonAnL, 725
MonAnS, 755

MPC10x10, 1250
OpAnL, 394
OpAnS, 415
Overview, 233
PIDConL, 1034
PIDConR, 1117
PIDConS, 1079
PIDStepL, 1158
Specifying the reaction to exiting local mode
With the Feature parameter, 189
Specifying warning times for control functions at motors and valves
VlvAnL, 1830
Split-range characteristics, 2833
Split-range control, 876, 916, 970, 1017, 1094, 1141
ConPerMon, 852
Splitting of the output signal of a controller
SplRange, 1218
SplRange
Area of application, 1216
Block diagram, 1223
Configuration, 1217
Error handling, 1221
Functions, 1218
How it works, 1216
I/Os, 1222
Messaging, 1221
Object name, 1216
Operating modes, 1217
Overview of error numbers, 1221
Splitting of the output signal of a controller, 1218
Startup characteristics, 1217
Status word allocation, 1217
SqrRoot
Area of application, 2303
Block diagram, 2307
Configuration, 2303
Error handling, 2305
Forming the signal status for blocks, 2304
Functions, 2304
How it works, 2303
I/Os, 2306
Messaging, 2305
Object name, 2303
Operating modes, 2304
Startup characteristics, 2303
Status word allocation, 2303
Standard monitoring functions, 92
Standard view
DoseL, 1350
FM controller, 294, 298, 302, 306
GainSched, 967
MonAnS, 763
MotSpdL, 1591
OpStations, 467
PIDConL, 1061
PIDConR, 1061
PIDConS, 1061
PIDStepL, 1185, 1189
ShrdResL, 1620
ShrdResS, 1642
VlvAnL, 1852
VlvMotL, 1792
VlvPosL, 1930
Standard view of CntOhSc, 2166
Start readiness, 59
Startup characteristics
AbsR, 2207
Add04, 2210
Add08, 2215
And04, 2399
And08, 2404
AssetM, 2778
AV, 705
Average, 2221
CompAn02, 2319
ConPerMon, 843
CountOh, 2100
CountScL, 2077
DeadTime, 2227
Derivative, 2235
Div02, 2240
DoseL, 1306
Event, 2017
Event16Ts, 2058
EventNck, 2030
EventTs, 2043
FbAnIn, 2467
FbAnOu, 2477
FbAnTot, 2488
FbDiIn, 2496
FbDiOu, 2506
FbEnMe, 2529
FbSwtMMS, 2542
Feature parameter, 148
FirstIn, 2013
FlipFlop, 2411
FlowCorr, 2247
FmCont, 877
FmTemp, 917
GainSched, 961
Integral, 2254
Intlk02, 1950

Intlk04, 1962
Intlk08, 1976
Intlk16, 1992
KalFilt, 1281
Lag, 2263
Limit, 2326
MeanTime, 2269
ModPreCon, 973
MonAnL, 715
MonAnS, 750
MonDi08, 816
MonDiL, 771
MonDiS, 797
MotL, 1376
MotRevL, 1440
MotS, 1415
MotSpdCL, 1488
MotSpdL, 1556
MPC10x10, 1240
MSTIn, 2738
MSTOu, 2742
Mul04, 2276
Mul08, 2281
MuxAn03, 2330
MuxAn08, 2338
MuxMST, 2765
MuxST, 2770
OpAnL, 389
OpAnS, 412
OpDi01, 426
OpDi03, 441
OpStations, 458
OpTrig, 470
Or04, 2418
Or08, 2422
Pcs7AITC, 2553
Pcs7AnIn, 2566
Pcs7AnOu, 2579
Pcs7Cnt2, 2639
Pcs7Cnt3, 2647
Pcs7DiCnt, 2589
Pcs7DiIn, 2597
Pcs7DiIT, 2606
Pcs7DiOu, 2617
Pcs7HaAI, 2666
Pcs7HaAO, 2673
PIDConL, 1019
PIDConR, 1096
PIDConS, 1070
PIDStepL, 1142
Polygon, 2289
PowerXY, 2314
Psc7Cnt1, 2626
RateLim, 2345
Ratio, 1197
Rcv_AnaVal, 2813
Rcv_DigVal, 2806
RedAn02, 2352
RedDi02, 2431
SelA02In, 2357
SelA16In, 2362
SelAn16L, 2381
SelD02In, 2436
ShrdResL, 1604
ShrdResS, 1627
Smooth, 2298
Snd_DigVal, 2802, 2810
Specifying, 148
SplRange, 1217
SqrRoot, 2303
STIn, 2730
STOu, 2734
StrctCom, 2441
StrctDeC, 2446
StruAnIn, 2706
StruAnOu, 2710
StruDiIn, 2714
StruDiOu, 2718
StruScIn, 2722
StruScOu, 2726
StruToBy, 2701
Sub02, 2308
TimerP, 2173
TimeTrig, 2182
TotalL, 2130
Trigger, 2451
Vlv2WayL, 1647
VlvAnL, 1809
VlvL, 1691
VlvMotL, 1754
VlvPosL, 1876
VlvS, 1728
XOr04, 2457
Startup characteristics over Trigger block, 75
StateMap
Description, 2750
Description of, 2750
Error handling, 2751
I/Os, 2751
Messaging, 2751
StateMap modes, 2751
Static and dynamic errors, 104
Static signal, 57
Stationary reference operating point, 846

Status diagram
DoseL, 1312
Status word allocation
AbsR, 2207
Add04, 2210
Add08, 2215
And04, 2399
And08, 2404
AssetM, 2778
AV, 706
Average, 2221
CompAn02, 2320
ConPerMon, 844
CountScL, 2078
DeadTime, 2228
Derivative, 2235
Div02, 2240
DoseL, 1306
Event, 2018
Event16Ts, 2059
EventNck, 2031
EventTs, 2043
FbAnIn, 2467
FbAnOu, 2477
FbDiIn, 2496
FbDiOu, 2506
FbEnMe, 2529
FirstIn, 2013
FlipFlop, 2411
FlowCorr, 2247
FmCont, 877
FmTemp, 917
GainSched, 961
Integral, 2254
Intlk02, 1950
Intlk04, 1962
Intlk08, 1976
Intlk16, 1992
KalFilt, 1281
Lag, 2263
Limit, 2326
MeanTime, 2269
ModPreCon, 974
MonAnL, 716
MonAnS, 750
MonDi08, 816
MonDiL, 771
MonDiS, 797
MotL, 1376
MotRevL, 1440
MotS, 1415
MotSpdCL, 1489
MotSpdL, 1556
MPC10x10, 1240
MSTIn, 2738
MSTOu, 2742
Mul04, 2276
Mul08, 2282
MuxAn03, 2330
MuxAn08, 2338
MuxMST, 2765
MuxST, 2770
OpAnL, 389
OpAnS, 412
OpDi01, 426
OpDi03, 441
OpStations, 459
OpTrig, 470
Or04, 2418
Or08, 2423
Pcs7AITC, 2554
Pcs7AnIn, 2566
Pcs7AnOu, 2579
Pcs7Cnt2, 2639
Pcs7Cnt3, 2647
Pcs7DiCnt, 2589
Pcs7DiIn, 2598
Pcs7DiIT, 2607
Pcs7DiOu, 2618
Pcs7HaAI, 2666
Pcs7HaAO, 2673
PIDConL, 1019
PIDConR, 1097
PIDConS, 1070
PIDStepL, 1143
Polygon, 2289
PowerXY, 2314
Psc7Cnt1, 2626
RateLim, 2345
Ratio, 1197
RedAn02, 2352
RedDi02, 2431
SelA02In, 2357
SelA16In, 2363
SelAn16L, 2381
SelD02In, 2436
ShrdResL, 1605
ShrdResS, 1628
Smooth, 2298
SplRange, 1217
SqrRoot, 2303
STIn, 2730
STOu, 2734
StrctCom, 2441

StrctDeC, 2446
StruAnIn, 2706
StruAnOu, 2710
StruDiIn, 2714
StruDiOu, 2718
StruScIn, 2722
StruScOu, 2726
StruToBy, 2701
Sub02, 2309
TimerP, 2174
TotalL, 2131
Trigger, 2452
Vlv2WayL, 1648
VlvAnL, 1809
VlvL, 1692
VlvMotL, 1755
VlvS, 1729
XOr04, 2457
Status word assignment
VlvPosL, 1877
Step control mode for the speed change
MotSpdL, 1570
Step controller, 56, 876, 882, 916, 922
STIn
Area of application, 2730
Block diagram, 2733
Configuration, 2730
Error handling, 2731
Functions, 2731
How it works, 2730
I/Os, 2732
Messaging, 2732
Object name, 2730
Operating modes, 2731
Startup characteristics, 2730
Status word allocation, 2730
Stochastic characteristics, 841, 847, 2875
Stopping integration
Integral, 2256
Stopping the caculation of the ramp output value
RateLim, 2347
Stopping the calculation of mean values
MeanTime, 2270
STOu
Area of application, 2734
Block diagram, 2737
Configuration, 2734
Error handling, 2735
Functions, 2735
How it works, 2734
I/Os, 2736
Messaging, 2736
Object name, 2734
Operating modes, 2735
Startup characteristics, 2734
Status word allocation, 2734
StrctCom
Area of application, 2441
Block diagram, 2444
Configuration, 2441
Error handling, 2443
Functions, 2442
How it works, 2441
I/Os, 2443
Messaging, 2443
Object name, 2441
Operating modes, 2442
Startup characteristics, 2441
Status word allocation, 2441
StrctDeC
Area of application, 2446
Block diagram, 2449
Configuration, 2446
Error handling, 2448
Functions, 2447
How it works, 2446
I/Os, 2448
Messaging, 2448
Object name, 2446
Operating modes, 2447
Startup characteristics, 2446
Status word allocation, 2446
StrgToBy
Block diagram, 2704
Error handling, 2703
Functions, 2702
I/Os, 2703
Messaging, 2703
Operating modes, 2702
StruAnIn
Area of application, 2706
Block diagram, 2709
Configuration, 2706
Error handling, 2707
Functions, 2707
How it works, 2706
I/Os, 2708
Messaging, 2708
Modes, 2707
Object name, 2706
Startup characteristics, 2706
Status word allocation, 2706
StruAnOu
Area of application, 2710

Block diagram, 2713
Configuration, 2710
Error handling, 2711
Functions, 2711
How it works, 2710
I/Os, 2712
Messaging, 2712
Modes, 2711
Object name, 2710
Startup characteristics, 2710
Status word allocation, 2710
Structure segmentation at controllers
FmCont, 887
FmTemp, 927
PIDConL, 1028
PIDStepL, 1153
StruDiIn
Area of application, 2714
Block diagram, 2717
Configuration, 2714
Error handling, 2715
Functions, 2715
How it works, 2714
I/Os, 2716
Messaging, 2716
Modes, 2715
Object name, 2714
Startup characteristics, 2714
Status word allocation, 2714
StruDiOu
Area of application, 2718
Block diagram, 2721
Configuration, 2718
Error handling, 2719
Functions, 2719
How it works, 2718
I/Os, 2720
Messaging, 2720
Modes, 2719
Object name, 2718
Startup characteristics, 2718
Status word allocation, 2718
StruScIn
Area of application, 2722
Block diagram, 2725
Configuration, 2722
Error handling, 2723
Functions, 2723
How it works, 2722
I/Os, 2724
Messaging, 2724
Object name, 2722
Operating modes, 2723
Startup characteristics, 2722
Status word allocation, 2722
StruScOu
Area of application, 2726
Block diagram, 2729
Configuration, 2726
Error handling, 2727
Functions, 2727
How it works, 2726
I/Os, 2728
Messaging, 2728
Object name, 2726
Operating modes, 2727
Startup characteristics, 2726
Status word allocation, 2726
StruToBy
Area of application, 2701
Configuration, 2701
How it works, 2701
Object name, 2701
Startup characteristics, 2701
Status word allocation, 2701
Sub02
Area of application, 2308
Block diagram, 2312
Configuration, 2308
Error handling, 2310
Forming the signal status for blocks, 2309
Functions, 2309
How it works, 2308
I/Os, 2311
Messaging, 2310
Object name, 2308
Operating modes, 2309
Startup characteristics, 2308
Status word allocation, 2309
SumMsgAct group display for limit monitoring
CSF and ExtMsgx, 708, 719, 753, 850, 885, 925, 1025, 1106, 1150, 1319, 1499, 1830, 2080, 2133
Suppress messages using the MsgLock parameter
VlvPosL, 1894
Suppressing messages using the MsgLock parameter
ConPerMon, 856
CountOh, 2103
CountScL, 2081
DoseL, 1320
Event, 2021
Event16Ts, 2063
EventNck, 2033
EventTs, 2046

FmCont, 891
FmTemp, 933
MonAnL, 719
MonAnS, 753
MonDi08, 820
MonDiL, 777
MonDiS, 801
MotL, 1383, 1420
MotRevLL, 1449
MotSpdCL, 1499
MotSpdL, 1565
OpAnL, 391
PIDConL, 1033
PIDConR, 1116
PIDConS, 1079
PIDStepL, 1158
TotalL, 2133
Vlv2WayL, 1655
VlvAnL, 1822
VlvL, 1700
VlvMotL, 1764
Suppression and reporting of signal flutter
MonDiL, 774
Switching the limiting function on or off
RateLim, 2347
Switchover with P step, 81
Switchover without P step, 81

## T

Target setpoint, 131, 133
Time characteristics
Rcv_DigVal, 2806
Snd_DigVal, 2802, 2810, 2813
Time delay
MotRevL, 1448
VlvMotL, 1763
Time delay after changing direction of rotation or restart
MotSpdCL, 1498
Time delay after restart
MotL, 1385
MotSpdL, 1564
Time response
CountOh, 2100
CountScL, 2077
TotalL, 2130
Time stamp, 231
Pcs7DiIT, 2608
Time stamp as associated value of a message
Event16Ts, 2063
EventTs, 2046
TimerP
Area of application, 2173
Block diagram, 2180
Configuration, 2173
Error handling, 2177
Forming the signal status for blocks, 2177
Functions, 2174
How it works, 2173
I/Os, 2178
Initialization, 2176
Messaging, 2178
Object name, 2173
Operating modes, 2174
Overview of error numbers, 2177
Resetting the Out and TimeRemaining output parameters, 2176
Setting the time, 2176
Specifying the method of operation, 2175
Startup characteristics, 2173
Status word allocation, 2174
Time-stamped messages
DoseL, 1326
FmCont, 892
FmTemp, 933
MonAnL, 725
MonDi08, 821
MonDiL, 778
MotL, 1390
MotRevL, 1457
MotSpdCL, 1510
MotSpdL, 1574
PIDConL, 1034
PIDConR, 1117
PIDStepL, 1159
Vlv2WayL, 1660
VlvAnL, 1830
VlvL, 1703
VlvMotL, 1773
VlvPosL, 1906
TimeTrig
Area of application, 2181
Block diagram, 2197
Block icon, 2205
Configuration, 2181
Error handling, 2191
Functions, 2184
How it works, 2181
I/Os, 2192
Messaging, 2192
Operating modes, 2183
Overview of error numbers, 2191
Parameter view, 2201

Periodic output of a trigger, 2184
Preview, 2203
Standard view, 2198
Startup characteristics, 2182
Torque monitoring
VlvMotL, 1765
VlvPosL, 1895
TotalL
Area of application, 2123
Associated values, 2139
Block diagram, 2146
Block icon, 2154
Bypass function, 2134
Configurable reactions using the Feature parameter, 2135
Configuration, 2130
Error handling, 2138
Forming the signal status for blocks, 2134
Functions, 2133
How it works, 2123
I/Os, 2140
Limit monitoring of the count value, 2133
Limit view, 2150
Messaging, 2139
Object name, 2123
Opening additional faceplates, 2133
Operating modes, 2132
Operator permissions, 2135
Overview of error numbers, 2138
Parameter view, 2151
Preview, 2153
Process messages, 2139
Readback of the most recently calculated sum, 2136
Release for maintenance, 2137
Selecting a unit of measure, 2133
Setting the summing/integrating value to the default, 2137
SIMATIC BATCH functionality, 2137
Simulating signals, 2134
Standard view, 2147
Startup characteristics, 2130
Status word allocation, 2131
Suppressing messages using the MsgLock parameter, 2133
Time response, 2130
Tracking and limiting a manipulated variable
FmCont, 884
FmTemp, 924
ModPreCon, 977
MPC10x10, 1243
PIDConL, 1024
PIDConR, 1105
PIDConS, 1074
PIDStepL, 1149
VlvAnL, 1818
Tracking setpoint in manual mode
FmCont, 885
FmTemp, 925
Tracking the setpoint, 222
PIDConL, 1025
PIDConR, 1107
PIDConS, 1075
PIDStepL, 1151
Tracking values
Integral, 2255
Trajectory, 2884
Transmission of messages
FbSwtMMS, 2542
Trend control, 2850
Trend plotter, 842
Trend view, 341
ModPreCon, 1013
MPC 10x10, 1273
Trend view of MotSpdCl, 1550
Trigger
Area of application, 2451
Block diagram, 2455
Configuration, 2451
Error handling, 2453
Forming the signal status for blocks, 2453
Functions, 2453
How it works, 2451
Messaging, 2454
Object name, 2451
Operating modes, 2452
Startup characteristics, 2451
Status word allocation, 2452
Truth table
XOr04, 2456


## U


Unit of measure, 241
Use output point for the manipulated variable calculation
PIDConR, 1111
User-configured message classes, 45
Using a manipulated variable ramp
VlvAnL, 1818
Using setpoint ramp
FmCont, 885
FmTemp, 925
OpAnL, 391

Using the manipulated variable ramp, 133
Using the setpoint ramp, 131

## V

Value application delay
  Pcs7AITC, 2556
  Pcs7AnIn, 2569
Valve
  Warning times, 56
Views of CntOhSc, 2165
Vlv2WayL
  Area of application, 1647
  Associated values, 1664
  Block diagram, 1676
  Block icon, 1688
  Button labels, 1660
  Configurable reactions using the Features parameter, 1659
  Configuration, 1647
  Considering bad quality of automatic commands or external values, 1657
  Defining valve positions for individual valves, 1653
  Disabling feedback, 1655
  Disabling interlocks, 1656
  Displaying auxiliary values, 1660
  Error handling, 1660
  Feedback monitoring, 1654
  Forcing operating modes, 1655
  Forming the group status for interlocks, 1657
  Forming the signal status for blocks, 1657
  Functions, 1653
  Generating instance-specific messages, 1658
  Group error, 1656
  How it works, 1647
  I/Os, 1664
  Instance-specific messages, 1663
  Interlocks, 1656
  Invalid or not plausible input commands, 1661
  Messaging, 1663
  Mode switchover error, 1661
  Neutral position, 1654
  Object name, 1647
  Opening additional faceplates, 1658
  Operating modes, 1651
  Operator permissions, 1658
  Output signal as a pulse signal or static signal, 1653
  Outputting a signal for start readiness, 1656
  Overview of error numbers, 1661
  Preview, 1683
  Process control fault, 1663
  Release for maintenance, 1657
  Resetting the block in case of interlocks or errors, 1656
  Selecting a unit of measure, 1658
  SIMATIC BATCH functionality, 1660
  Simulating signals, 1655
  Specify warning times for control functions, 1654
  Standard view, 1677
  Startup characteristics, 1647
  Status word allocation, 1648
  Suppressing messages using the MsgLock parameter, 1655
  Time-stamped messages, 1660
VlvAnL
  Actuator active information, 1829
  Alarm delays with one time value per limit pair, 1826
  Area of application, 1808
  Associated values, 1834
  Block diagram, 1850
  Block icon, 1872
  Button labels, 1831
  Configurable reactions using the Feature parameter, 1825
  Configuration, 1808
  Considering bad quality of automatic commands or external values, 1820
  Digital feedback from the readback value, 1826
  Disabling feedback, 1830
  Disabling of limits, 1823
  Displaying auxiliary values, 1826
  Error handling, 1832
  Feedback monitoring, 1820
  Forcing operating modes, 1820
  Forming the group status for interlocks, 1819
  Forming the signal status for blocks, 1819
  Functions, 1816
  General function MV difference, 1829
  Generating instance-specific messages, 1823
  Generation of manipulated variables, 1828
  Gradient limiting of the manipulated variable, 1818
  Group error, 1817
  How it works, 1808
  I/Os, 1835
  Interlocks, 1817
  Invalid input signals, 1832
  Limit monitoring of manipulated variable and error signal, 1830
  Limit view, 1863, 1935

Manipulated variable difference generation and dead band, 1829
Messaging, 1833
Mode switchover error, 1832
Monitoring the feedback for the auxiliary valve, 1822
Neutral position, 1823
Object name, 1808
Opening additional faceplates, 1817
Operating modes, 1813
Operator permissions, 1827
Outputting a signal for start readiness, 1818
Overview of error numbers, 1832
Parameter view, 1871
Preview, 1865
Process control fault, 1833
Process messages, 1834
Resetting the block in case of interlocks or errors, 1817
Selecting a unit of measure, 1823
SIMATIC BATCH functionality, 1818
Simulating signals, 1818
Specifying warning times for control functions at motors and valves, 1830
Standard view, 1857
Startup characteristics, 1809
Status word allocation, 1809
Suppressing messages using the MsgLock parameter, 1822
Time-stamped messages, 1830
Tracking and limiting a manipulated variable, 1818
Using a manipulated variable ramp, 1818
VlvL
Area of application, 1691
Associated values, 1706
Block diagram, 1715
Block icon, 1725
Button labels, 1703
Configurable reactions using the Feature parameter, 1701
Configuration, 1691
Considering bad quality of automatic commands or external values, 1698
Disabling feedback, 1699
Disabling interlocks, 1697
Displaying auxiliary values, 1703
Error handling, 1703
Feedback monitoring, 1699
Forcing operating modes, 1698
Forming the group status for interlocks, 1698
Forming the signal status for blocks, 1698
Functions, 1696
Generating instance-specific messages, 1700
Group error, 1697
How it works, 1691
I/Os, 1707
Instance-specific messages, 1706
Interlocks, 1697
Invalid or not plausible input commands, 1704
Messaging, 1705
Mode switchover error, 1704
Neutral position, 1700
Object name, 1691
Opening additional faceplates, 1696
Operating modes, 1694
Operator permissions, 1696
Output signal as a pulse signal or static signal, 1703
Outputting a signal for start readiness, 1698
Overview of error numbers, 1704
Preview, 1721
Process control fault, 1706
Release for maintenance, 1700
Resetting the block in case of interlocks, 1697
Selecting a unit of measure, 1700
SIMATIC BATCH functionality, 1703
Simulating signals, 1700
Specify warning times for control functions, 1700
Standard view, 1717
Startup characteristics, 1691
Status word allocation, 1692
Suppressing messages using the MsgLock parameter, 1700
Time-stamped messages, 1703
VlvL";"Disabling feedback"}{"Disabling feedback";"VlvL, 1386, 1451, 1504, 1699
VlvMotL
Area of application, 1754
Associated values, 1778
Block diagram, 1791
Block icon, 1805
Button labels, 1773
Configurable reactions using the Feature parameter, 1772
Configuration, 1754
Considering bad quality of automatic commands or external values, 1768
Disabling feedback, 1769
Disabling interlocks, 1766
Disabling of limits, 1771
Displaying auxiliary values, 1773
Error handling, 1774
Feedback monitoring, 1768

Forcing operating modes, 1768
Forming the group status for interlocks, 1767
Forming the signal status for blocks, 1767
Functions, 1762
Generating instance-specific messages, 1771
Group error, 1766
How it works, 1754
I/Os, 1778
Instance-specific messages, 1777
Interlocks, 1764
Invalid or not plausible input commands, 1775
Limit monitoring of an additional analog value, 1764
Limit monitoring with hysteresis, 1764
Messaging, 1777
Mode switchover error, 1775
Motor protection function, 1764
Neutral position, 1770
Object name, 1754
Opening additional faceplates, 1762
Operating modes, 1759
Operator permissions, 1762
Output signal as a pulse signal or static signal, 1771
Outputting a signal for start readiness, 1767
Overview of error numbers, 1775
Parameter view, 1797
Preview, 1799
Process control fault, 1777
Rapid stop, 1764
Release for maintenance, 1770
Reset of motor control after valve end position has been reached, 1765
Resetting the block in case of interlocks, 1766
Sealing the valve, 1765
Selecting a unit of measure, 1770
SIMATIC BATCH functionality, 1773
Simulating signals, 1770
Specify warning times for control functions, 1770
Startup characteristics, 1754
Status word allocation, 1755
Suppressing messages using the MsgLock parameter, 1764
Time delay after changing direction of rotation or restart, 1763
Time-stamped messages, 1773
Torque monitoring, 1765
Tracking the valve position after a motor stop or after a startup, 1769

VlvPosL
Associated values, 1911, 1912
Block diagram, 1928
Block icon, 1947
Configurable reactions using the Feature parameter, 1905
Configuration, 1876
Considering bad quality of automatic commands or external values, 1899
Control system fault, 1910
Disable feedback, 1906
Disable interlocks, 1897
Display auxiliary values, 1906
Error handling, 1908
Feedback monitoring, 1900
Force operating states, 1900
Forming the group status for interlocks, 1898
Forming the signal status for blocks, 1898
Functions, 1884
Generate instance-specific messages, 1904
Group error, 1898
How it works, 1876
I/Os, 1912
Instance-specific messages, 1911
Interlocks, 1894
Invalid or not plausible input commands, 1909
Labeling of buttons, 1907
Limit monitoring for additional analog value, 1893
Limit monitoring with hysteresis, 1893
Message characteristics, 1910
Mode switchover error, 1908
Motor protection function, 1895
Neutral position, 1903
Object name, 1876
Opening additional faceplates, 1891
Operating modes, 1882
Operator permissions, 1891
Output signal as pulse signal or static signal, 1903
Outputting a signal for start readiness, 1898
Overview of error numbers, 1908
Parameter view, 1937
Preview, 1941
Rapid stop, 1895
Release for maintenance, 1902
Resetting the block in case of interlocks, 1897
Sealing the valve, 1896
Select unit of measure, 1903
SIMATIC BATCH functionality, 1906
Simulating signals, 1903
Specify warning times for control functions, 1902
Startup characteristics, 1876
Subentry, 1876

Suppress messages using the MsgLock parameter, 1894
Time-stamped messages, 1906
Torque monitoring, 1895
VlvS
Area of application, 1728
Associated values, 1739
Block diagram, 1745
Block icon, 1751
Configurable reactions using the Feature parameter, 1735
Configurable reactions using the Feature2 parameter, 1736
Configuration, 1728
Considering bad quality of automatic commands or external values, 1735
Disabling interlocks, 1733
Error handling, 1736
Feedback monitoring, 1735
Forming the group status for interlocks, 1734
Forming the signal status for blocks, 1734
Functions, 1732
Generating instance-specific messages, 1735
Group error, 1734
How it works, 1728
I/Os, 1739
Instance-specific messages, 1738
Interlocks, 1733
Invalid input signals, 1737
Messaging, 1738
Mode switchover error, 1737
Neutral position, 1734
Object name, 1728
Opening additional faceplates, 1732
Operating modes, 1731
Operator permissions, 1732
Overview of error numbers, 1737
Preview, 1749
Process control fault, 1738
Release for maintenance, 1735
Resetting the block in case of interlocks, 1733
SIMATIC BATCH functionality, 1736
Simulating signals, 1735
Standard view, 1746
Startup characteristics, 1728
Status word allocation, 1729

## W

Warning signals, 56

## X

XOr04
Area of application, 2456
Block diagram, 2460
Configuration, 2456
Error handling, 2458
Forming the signal status for blocks, 2458
Functions, 2457
How it works, 2456
I/Os, 2459
Messaging, 2458
Object name, 2456
Operating modes, 2457
Startup characteristics, 2457
Status word allocation, 2457
Truth table, 2456