

Festo PA-Toolkit Manual

Date
24.08.2022

Author
BUNK

Version
2.0.5.0

1	General information	11
1.1	Version management	11
1.1.1	Change management	11
1.1.2	Release 1.0.1.1, 1.0.1.2, 1.0.1.3 and 2.0.5.0	12
2	Licence for PA Toolkit.....	13
2.1	Brief description	13
2.2	Demo licence	13
2.3	Purchasing and installing a licence.....	14
2.4	Symbol.....	14
2.4.1	hLicenseDefault.....	14
2.5	Faceplate.....	15
2.5.1	Main.....	15
3	Analogue value display.....	16
3.1	AnaView function block.....	16
3.1.1	Brief description	16
3.1.2	Basic functions	16
3.1.3	Functions.....	16
3.1.4	FB-specific worst quality code (WQC)	18
3.1.5	Persistent variables	18
3.1.6	Symbols	19
3.2	AnaMon function block	22
3.2.1	Brief description	22
3.2.2	Extended functions	22
3.2.3	FB-specific worst quality code (WQC)	23
3.2.4	Persistent variables	24
3.2.5	Symbols	25
3.2.6	Faceplates	27
4	Analogue value operation	29
4.1	AnaMan function block	29
4.1.1	Brief description	29
4.1.2	Basic functions	29
4.1.3	Functions.....	30
4.1.4	FB-specific worst quality code (WQC)	32
4.1.5	Persistent variables	32
4.1.6	Symbols	33
4.2	AnaManInt function block	34
4.2.1	Brief description	34
4.2.2	Extended functions	34
4.2.3	FB-specific worst quality code (WQC)	34
4.2.4	Source mode	35
4.2.5	Persistent variables	36
4.2.6	Symbols	37
4.2.7	Faceplates	38

5	Digital value display	40
5.1	DIntView function block	40
5.1.1	Basic functions	40
5.1.2	Functions.....	40
5.1.3	FB-specific worst quality code (WQC)	42
5.1.4	Persistent variables	42
5.1.5	Symbols	43
5.2	DIntMon function block.....	46
5.2.1	Brief description	46
5.2.2	Extended functions	46
5.2.3	FB-specific worst quality code (WQC)	47
5.2.4	Persistent variables	48
5.2.5	Symbols	49
5.2.6	Faceplates	51
6	Digital value operation.....	53
6.1	DIntMan function block.....	53
6.1.1	Brief description	53
6.1.2	Basic functions	53
6.1.3	Functions.....	54
6.1.4	FB-specific worst quality code (WQC)	56
6.1.5	Persistent variables	56
6.1.6	Symbols	57
6.2	DIntManInt function block.....	58
6.2.1	Brief description	58
6.2.2	Extended functions	58
6.2.3	FB-specific worst quality code (WQC)	58
6.2.4	Source mode	59
6.2.5	Persistent variables	60
6.2.6	Symbols	62
6.2.7	Faceplates	63
7	Binary value display	64
7.1	BinView function block	64
7.1.1	Brief description	64
7.1.2	Basic functions	64
7.1.3	Functions.....	64
7.1.4	FB-specific worst quality code (WQC)	65
7.1.5	Symbols	65
7.2	BinMon function block	67
7.2.1	Brief description	67
7.2.2	Extended functions	67
7.2.3	FB-specific worst quality code (WQC)	68
7.2.4	Persistent variables	68
7.2.5	Symbols	70
7.2.6	Faceplates	71

8	Binary operations.....	73
8.1	BinMan function block	73
8.1.1	Brief description	73
8.1.2	Basic functions	73
8.1.3	Functions.....	74
8.1.4	FB-specific worst quality code (WQC)	75
8.1.5	Persistent variables	75
8.1.6	Symbols	76
8.2	BinManInt function block	77
8.2.1	Brief description	77
8.2.2	Extended functions	77
8.2.3	FB-specific worst quality code (WQC)	77
8.2.4	Source mode	77
8.2.5	Persistent variables	79
8.2.6	Symbols	80
8.2.7	Faceplates	81
9	Character string display	82
9.1	StringView function block	82
9.1.1	Brief description	82
9.1.2	Functions.....	82
9.1.3	FB-specific worst quality code (WQC)	82
9.1.4	Persistent variables	82
9.1.5	Symbols	83
9.1.6	Faceplates	83
10	Controller.....	84
10.1	PIDCtrl function block	84
10.1.1	Brief description	84
10.1.2	Basic functions	84
10.1.3	Functions.....	85
10.1.4	Extended functions	87
10.1.5	FB-specific worst quality code (WQC)	88
10.1.6	Source mode	89
10.1.7	Operation mode.....	91
10.1.8	Persistent variables	93
10.1.9	Symbols	94
10.1.10	Faceplates	95
11	Bistable valves.....	97
11.1	BinVlv function block	97
11.1.1	Brief description	97
11.1.2	Basic functions	97
11.1.3	Functions.....	98
11.1.4	FB-specific worst quality code (WQC)	100
11.1.5	Operation mode.....	100
11.1.6	Persistent variables	102
11.1.7	Symbols	103
11.2	MonBinVlv function block	105

11.2.1	Brief description	105
11.2.2	Extended functions	105
11.2.3	FB-specific worst quality code (WQC)	107
11.2.4	Persistent variables	107
11.2.5	Symbols	108
11.2.6	Faceplates	110

12 Controllable valves 112

12.1	AnaVlv function block	112
12.1.1	Brief description	112
12.1.2	Basic functions	112
12.1.3	Functions.....	113
12.1.4	FB-specific worst quality code (WQC)	116
12.1.5	Source mode	116
12.1.6	Operation mode.....	118
12.1.7	Persistent variables	120
12.1.8	Symbols	121
12.2	MonAnaVlv function block	124
12.2.1	Brief description	124
12.2.2	Extended functions	124
12.2.3	FB-specific worst quality code (WQC)	127
12.2.4	Persistent variables	127
12.2.5	Symbols	129
12.2.6	Faceplates	131

13 Bistable motors..... 133

13.1	BinDrv function block.....	133
13.1.1	Brief description	133
13.1.2	Basic functions	133
13.1.3	Functions.....	134
13.1.4	FB-specific worst quality code (WQC)	137
13.1.5	Operation mode.....	137
13.1.6	Persistent variables	139
13.1.7	Symbols	140
13.2	MonBinDrv function block.....	142
13.2.1	Brief description	142
13.2.2	Extended functions	142
13.2.3	FB-specific worst quality code (WQC)	144
13.2.4	Persistent variables	144
13.2.5	Symbols	146
13.2.6	Faceplates	148

14 Controllable motors 150

14.1	AnaDrv function block.....	150
14.1.1	Brief description	150
14.1.2	Basic functions	150
14.1.3	Functions.....	151
14.1.4	FB-specific worst quality code (WQC)	155
14.1.5	Source mode	155

14.1.6	Operation mode.....	157
14.1.7	Persistent variables	159
14.1.8	Symbols	160
14.2	MonAnaDrv function block.....	162
14.2.1	Brief description	162
14.2.2	Extended functions	162
14.2.3	FB-specific worst quality code (WQC)	165
14.2.4	Persistent variables	165
14.2.5	Symbols	167
14.2.6	Faceplates	169
15	Totalizer.....	171
15.1	Totalizer function block.....	171
15.1.1	Brief description	171
15.1.2	Basic functions	171
15.1.3	Functions.....	172
15.1.4	FB-specific worst quality code (WQC)	174
15.1.5	Source mode	174
15.1.6	Operation mode.....	176
15.1.7	Persistent variables	177
15.1.8	Symbols	178
15.2	MonTotalizer function block.....	180
15.2.1	Brief description	180
15.2.2	Extended Functions	180
15.2.3	FB-specific worst quality code (WQC)	181
15.2.4	Persistent variables	181
15.2.5	Symbols	183
15.2.6	Faceplates	184
16	MPA positioner	186
16.1	DrvCtrl function block	186
16.1.1	Brief description	186
16.1.2	Basic functions	186
16.1.3	Functions.....	187
16.1.4	FB-specific worst quality code (WQC)	189
16.1.5	Source mode	190
16.1.6	Operation mode.....	191
16.1.7	Persistent variables	193
16.1.8	Symbols	195
16.1.9	Faceplates	196
17	Interlock display	199
17.1	LockView4 function block	199
17.1.1	Brief description	199
17.1.2	Basic functions	199
17.1.3	Functions.....	199
17.1.4	FB-specific worst quality code (WQC)	200
17.1.5	Persistent variables	200
17.1.6	Symbols	201

17.1.7	Faceplates	201
17.2	LockView8 function block	203
17.2.1	Brief description	203
17.2.2	Basic functions	203
17.2.3	Functions.....	203
17.2.4	Persistent variables	204
17.2.5	Symbols	205
17.2.6	Faceplates	205
17.3	LockView16 function block	207
17.3.1	Brief description	207
17.3.2	Basic functions	207
17.3.3	Functions.....	207
17.3.4	FB-specific worst quality code (WQC)	208
17.3.5	Persistent variables	208
17.3.6	Symbols	209
17.3.7	Faceplates	209
18	Services	212
18.1	SimpleSelfCompletingServiceControl	212
18.1.1	Brief description	212
18.1.2	Basic functions	213
18.1.3	Functions.....	214
18.1.4	Extended functions	217
18.1.5	Source mode	217
18.1.6	Operation mode.....	218
18.2	SimpleContinuousServiceControl.....	221
18.2.1	Brief description	221
18.2.2	Basic functions	222
18.2.3	Functions.....	222
18.2.4	Extended functions	226
18.2.5	Source mode	226
18.2.6	Operation mode.....	227
18.3	ServiceControl	230
18.3.1	Brief description	230
18.3.2	Basic functions	231
18.3.3	Functions.....	232
18.3.4	Source mode	235
18.3.5	Operation mode.....	236
18.4	Symbols	239
18.4.1	Description.....	239
18.5	Faceplates	240
18.5.1	Main.....	240
18.5.2	Operate	240
19	ServiceProcedures	241
19.1	ServiceProcedure	241
19.1.1	Brief description	241
19.1.2	Basic functions	241
19.1.3	Functions.....	241

20	Analogue service parameters.....	243
20.1	AnaConfigParam	243
20.1.1	Brief description	243
20.1.2	Basic functions	243
20.1.3	Functions.....	244
20.1.4	Source mode	245
20.1.5	Operation mode.....	247
20.2	AnaProcedureParam	249
20.2.1	Brief description	249
20.2.2	Basic functions	249
20.2.3	Functions.....	250
20.2.4	Source mode	251
20.2.5	Operation mode.....	253
20.3	AnaProcessValueIn	256
20.3.1	Brief description	256
20.3.2	Basic functions	256
20.3.3	Functions.....	257
20.4	AnaProcessValueOut	258
20.4.1	Brief description	258
20.4.2	Basic functions	258
20.4.3	Functions.....	259
20.4.4	FB-specific worst quality code (WQC)	261
20.4.5	Persistent variables	261
20.5	AnaReportValue.....	263
20.5.1	Brief description	263
20.5.2	Basic functions	263
20.5.3	Functions.....	264
20.5.4	FB-specific worst quality code (WQC)	266
20.5.5	Persistent variables	266
21	Binary service parameters.....	268
21.1	BinConfigParam	268
21.1.1	Brief description	268
21.1.2	Basic functions	268
21.1.3	Functions.....	269
21.1.4	Source mode	270
21.1.5	Operation mode.....	271
21.2	BinProcedureParam	274
21.2.1	Brief description	274
21.2.2	Basic functions	274
21.2.3	Functions.....	275
21.2.4	Source mode	276
21.2.5	Operation mode.....	277
21.3	BinProcessValueIn	280
21.3.1	Brief description	280
21.3.2	Basic functions	280
21.3.3	Functions.....	281
21.4	BinProcessValueOut	282
21.4.1	Brief description	282
21.4.2	Basic functions	282

21.4.3	Functions.....	283
21.4.4	FB-specific worst quality code (WQC)	283
21.5	BinReportValue	284
21.5.1	Brief description	284
21.5.2	Basic functions	284
21.5.3	Functions.....	285
21.5.4	FB-specific worst quality code (WQC)	286

22 Digital service parameters 287

22.1	DIntConfigParam	287
22.1.1	Brief description	287
22.1.2	Basic functions	287
22.1.3	Functions.....	288
22.1.4	Source mode	289
22.1.5	Operation mode.....	291
22.2	DIntProcedureParam.....	293
22.2.1	Brief description	293
22.2.2	Basic functions	293
22.2.3	Functions.....	294
22.2.4	Source mode	295
22.2.5	Operation mode.....	297
22.3	DIntProcessValueIn	300
22.3.1	Brief description	300
22.3.2	Basic functions	300
22.3.3	Functions.....	301
22.4	DIntProcessValueOut.....	302
22.4.1	Brief description	302
22.4.2	Basic functions	302
22.4.3	Functions.....	303
22.4.4	FB-specific worst quality code (WQC)	305
22.4.5	Persistent variables	305
22.5	DIntReportValue	307
22.5.1	Brief description	307
22.5.2	Basic functions	307
22.5.3	Functions.....	308
22.5.4	FB-specific worst quality code (WQC)	310
22.5.5	Persistent variables	310

23 Character string service parameters..... 312

23.1	StringConfigParam.....	312
23.1.1	Brief description	312
23.1.2	Basic functions	312
23.1.3	Functions.....	313
23.1.4	Source mode	315
23.1.5	Operation mode.....	316
23.2	StringProcedureParam.....	319
23.2.1	Brief description	319
23.2.2	Basic functions	319
23.2.3	Functions.....	320

23.2.4	Source mode	321
23.2.5	Operation mode.....	322
23.3	StringProcessValueOut	325
23.3.1	Brief description	325
23.3.2	Functions.....	326
23.4	StringReportValue	327
23.4.1	Brief description	327
23.4.2	Basic functions	327
23.4.3	Functions.....	328
24	Parameter blocks	329
24.1	Symbols	329
24.2	Faceplates Bool-Parameter	330
24.3	Faceplates Real-Parameter	331
24.4	Faceplates String-Parameter	332
25	Help functions and enumeration	333
25.1	Help functions	333
25.1.1	Scaler	333
25.2	Enumeration.....	334
25.2.1	WorstQualityCode enumeration	334
25.2.2	UnitSettings enumeration	334
25.2.3	ChannelAutOp enumeration.....	334
25.2.4	LockLogic enumeration	334
25.2.5	Command enumeration	335
25.2.6	SMState enumeration	335

1 General information

- The PA Toolkit has been implemented in accordance with the current MTP specification for function blocks and services (revision level: April 2020). The MTP exporter also functions with this revision level.
- All values are calculated accurately to 2 places after the decimal point. Discrepancies in further decimal places are software-related.
- Permanently linked values can no longer be changed via a faceplate.
- The variables for controlling the source and operation modes must not be permanently linked, because the handshake procedure would Otherwise, no longer function.
- Pointers are used to shift the faceplates. After an online change, the address ranges for pointers may be overridden and shifting may not function correctly anymore, for which reason online changes should be avoided.
- Floating point numbers with 5 places before and 2 places after the decimal are correctly displayed in the visualisation. 5 places are available for integer values. If more places are used, they might be cut off in the visualisation.
- Units of measure with large numbers of characters may be cut off in the faceplates for space reasons.

1.1 Version management

It must be assured that old software is not used with the controllers. Exception errors may Otherwise, occur due to the licencing process, which accesses functions from service pack 12.

1.1.1 Change management

Release	Changes and extensions
1.0.1.1	Full package with function blocks, symbols, faceplates, MTP-Exporter and style for visualisation
1.0.1.2	• Bugfix MTP-Exporter • new output parameter at the parameter function blocks • Bugfix Services • Bugfix Symbols
1.0.1.3	• Bugfix MTP-Exporter • Bugfix license • Bugfix Services • Bugfix documentation • English manual is available
2.0.5.0	• WIBU Licence • MTP Service Faceplate • Bugfixes

Tab. 1 Change management

1.1.2 Release 1.0.1.1, 1.0.1.2, 1.0.1.3 and 2.0.5.0

This release has been tested with an approved for the following versions:

Software/hardware	Version
CoDeSys	V3.5 SP12 Patch 6 pbF
Browser: Chrome	Version 81.0.4044.122 (Official Build) (64-Bit)
CPX-CEC-C1-V3	3.0.8.0.017190
CPX-CEC-S1-V3	3.0.8.0.017190
CPX-CEC-M1-V3	3.0.8.0.017190
CPX-E-CEC-M1-PN	1.1.22-73131e464.20190903
CPX-E-CEC-M1-EP	1.1.22-73131e464.20190903
CPX-E-CEC-C1-PN	1.1.22-73131e464.20190903
CPX-E-CEC-C1-EP	1.1.22-73131e464.20190903

Tab. 2 Release management

2 Licence for PA Toolkit

2.1 Brief description

A license is required for continuous operation of the PA Toolkit.

After downloading the PA Toolkit, the application can be implemented without any time limitations. If it's downloaded to a controller, checking is conducted to determine whether a license is available.

Each used function block needs an amount of Credits. There are two kinds of license. The Credits license permits function blocks up to the amount of installed Credits. These licenses can be installed multiple times to increase the total amount of installed Credits. A Flatrate license permits any amount of function blocks. When the amount of installed Credits passes a defined limit, then a Flatrate by Credits is activated.

2.2 Demo licence

If no license is available, the demo license is initialized, and the controller can be operated for **2 hours**. After 2 hours, the controller has to be restarted or the application has to be downloaded again. When the demo licence expires, all function blocks are set to a specific state. It's also no longer possible to make any settings via the visualisation. A corresponding message appears in the symbols. The states of each FB are described in the following table:

Function block	State
AnaView	V = VNormMin
AnaMon	V = VNormMin
AnaMan	VOut = SafeValue
AnaManInt	VOut = SafeValue
DIntView	V = VNormMin
DIntMon	V = VNormMin
DIntMan	VOut = SafeValue
DIntManInt	VOut = SafeValue
BinView	V = FALSE
BinMon	V = VAlmState
BinMan	VOut = SafeState
BinManInt	VOut = SafeState
StrView	TextOut = „“
PIDCtrl	Same performance as in offline mode
BinVlv	Same performance as in offline mode
MonBinVlv	Same performance as in offline mode, alarms continue to work
AnaVlv	Same performance as in offline mode
MonAnaVlv	Same performance as in offline mode, alarms continue to work
BinDrv	Same performance as in offline mode
MonBinDrv	Same performance as in offline mode, alarms continue to work
AnaDrv	Same performance as in offline mode
MonAnaDrv	Same performance as in offline mode, alarms continue to work
LockView4/LockView8/LockView16	Out = FALSE

Totalizer	Stop command is executed
MonTotalizer	Stop command is executed, alarms continue to work
Service	Service is set to hold
DrvCtrl	All valve outputs will be set to 0

Tab. 3 specific states of each function block

In order to continuously run the PA Toolkit on the controller, a licence must be purchased from Festo SE & Co.KG.

2.3 Purchasing and installing a licence

Licenses of kind Credits or Flatrate can be acquired via the Festo Appworld (<https://www.festo.com/de/en/pa-toolkit.html>).

Licenses are installed via the Festo Automation Suite. It is available via the Festo Support Portal (https://www.festo.com/net/en-gb_gb/SupportPortal/default.aspx?q=automation+suite).

Restart the device, if the license can not be found after installing.

To see the license, or the amount of needed and installed Credits proceed as follows:

- At least one PA Toolkit function block must be running in a task.
- Create a visualisation in CODESYS.
- Drag the hLicenseDefault symbol into the visualisation.
- Download and start the application.
- Open the faceplate by clicking the licence symbol.
- Checking to determine whether or not a license is available for the PA Toolkit can be conducted at any time by clicking the “Check Licence” button.

2.4 Symbol

2.4.1 hLicenseDefault

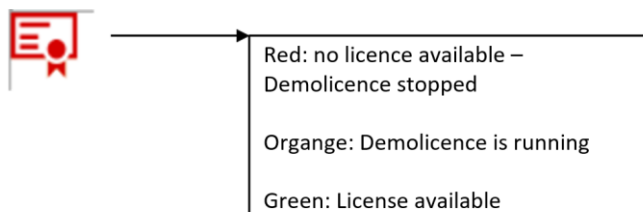


Fig. 1 License symbol

2.5 Faceplate

2.5.1 Main

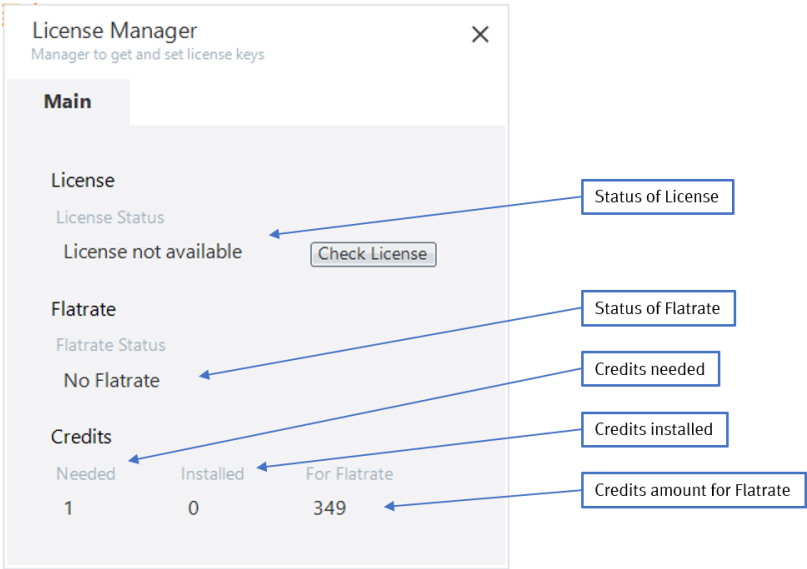


Fig. 2 License main faceplate

3 Analogue value display

3.1 AnaView function block

3.1.1 Brief description

The AnaView function block (FB) makes it possible to scale an analogue input value (e.g., from an analogue input card). This converted value is made available as an output for further processing.

3.1.2 Basic functions

3.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

3.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All possible descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 4 Worst quality code enumeration

3.1.3 Functions

3.1.3.1 Input values "FV" and "VSim" (field value or simulated value)

The input value is represented by the "FV" variable and is made available to the FB as an input. This input can be linked to a variable or an address of an analogue input card. Scaling of the input value is adjusted via standardisation values "FVNormMin" and "FVNormMax". Both of these values are also inputs at the FB.

If no input value is available, a value can be simulated at the FB via the "VSim" input. The simulation is activated via the "VSimEn" input bit (= TRUE). Output value "V" is simulated here directly. This means that the simulated value is not scaled. It's only checked to determine if the simulated values fall short of the "VNormMin" value or exceed the "VNormMax" value. If the limit values are not complied with, the corresponding limit value is read out as "V".

3.1.3.2 Output value “V”

Input value “FV” is converted to output value “V” with the help of the standardisation values (“FVNormMin”, “FVNormMax”, “VNormMin” and “VNormMax”) and is read out from the FB. As is also the case for the standardisation values for the input value, the standardisation values for “V” are available as inputs at the FB. If “V” exceeds or falls short of its standardisation values (“VNormMin” and “VNormMax”), the corresponding standardisation value is read as output “V”.

3.1.3.3 Conversion of input value “FV” to output value “V”

The standardisation values, which can be read into the FB as inputs, are required for the input as well as for the output in order to convert the input value to an output value.

The conversion formula is as follows:

$$V = \frac{VNormMax - VNormMin}{FVNormMax - FVNormMin} * (FV - FVNormMin) + VNormMin$$

Input: FVNormMin and FVNormMax

Output: VNormMin and VNormMax

3.1.3.4 Unit of measure for “V”

A unit of measure can be entered for the output value via the “VUnit” input variable at the FB. The “UnitSettings” enumeration is used in order to specify a unit of measure. The unit of measure is only used for displays and has no influence on the conversion.

3.1.3.5 Monitoring the standardisation values

As described above, the standardisation values limit the input and output values and are used for conversion. It must also be assured that the minimum standardisation values are not greater than the associated maximum standardisation values. Otherwise, there’s no guarantee that the conversion and the graphics in the visualisation will include sensible values.

Furthermore, the difference between “FVNormMax” and “FVNormMin” must not be zero, because it’s not possible to divide by zero. In this case zero is read out at output value “V”.

3.1.3.6 Scaling values for the visualisation

One Min and one Max variable (“VScMin” and “VScMax”) are used to set the display limits of each visual representation of the output value. These variables are used exclusively for the representation of graphic elements in the visualisation. If output “FullScEn” is set to TRUE, standardisation values “VNormMin” and “VNormMax” are used automatically as display limits. If the display limits are set incorrectly, they’re set automatically to the standardisation values. Refer to the section entitled “FB-specific worst quality code” in this regard.

3.1.4 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
				VScIMax < VScImin ⇒ VScIMax = VNormMax ⇒ VScImin = VNormMin
				VScIMax < VNormMin oder VScIMax > VNormMax ⇒ VScIMax = VNormMax
				VScImin < VNormMin oder VScImin > VNormMax ⇒ VScImin = VNormMin
				VScIMax – VScImin = 0 ⇒ VScIMax = VNormMax ⇒ VScImin = VNormMin
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 5 AnaView specific worst quality code

3.1.5 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
VNormMax	REAL	100	High Scale Value of Output Value (High limit of Output Value and scaling factor for V calculation)
VNormMin	REAL	0	Low Scale Value of Output Value (Low limit of Output Value and scaling factor for V calculation)

Tab. 6 AnaView persistent variables

3.1.6 Symbols

3.1.6.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Simulation symbol	Indicates when the output value is simulated. The output value is entered via “VSim”.	Appears when VSimEn = TRUE.
Output value	Output value “V” is displayed.	Max. 5 places before and 2 places after the decimal point are displayed.
Unit	U/M “VUnit” is displayed.	Only values from the “UnitSettings” enumeration are possible.
Nozzles	Nozzles for connecting measurement lines (solid lines) and logical lines (dashed lines).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
Bargraph	Graphically represents the selected limit values and the current output value. Limit values are only taken into consideration when limit value monitoring has been activated.	Limit values must be set correctly, because the bar graph will Otherwise, be incorrectly displayed (For example, limit value High must be greater than limit value Low). “VScIMin” and “VScIMax” scale the bar graph.
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 7 AnaView symbol description

3.1.6.2 hAnaViewDefault

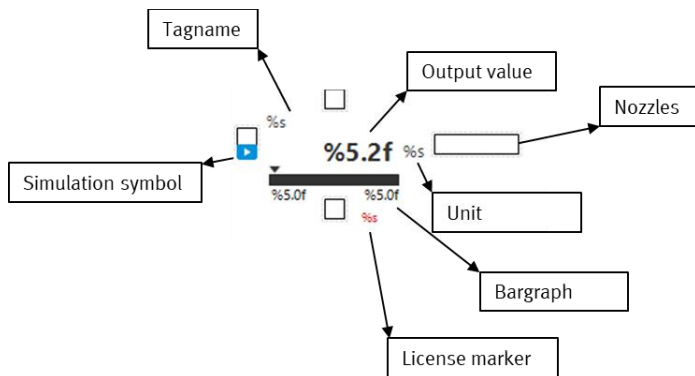


Fig. 3 Symbol hAnaViewDefault

3.1.6.3 hAnaViewValue

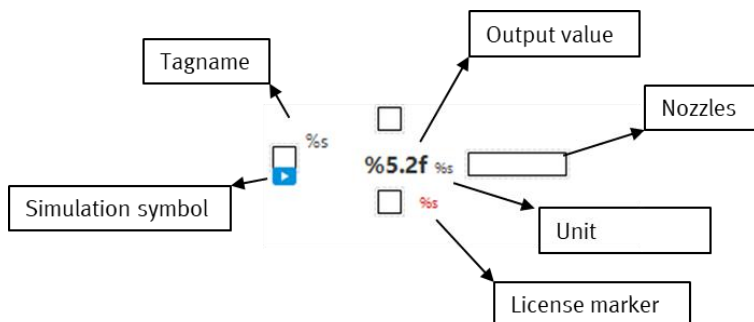


Fig. 4 Symbol hAnaViewValue

3.1.6.4 hAnaViewBargraph

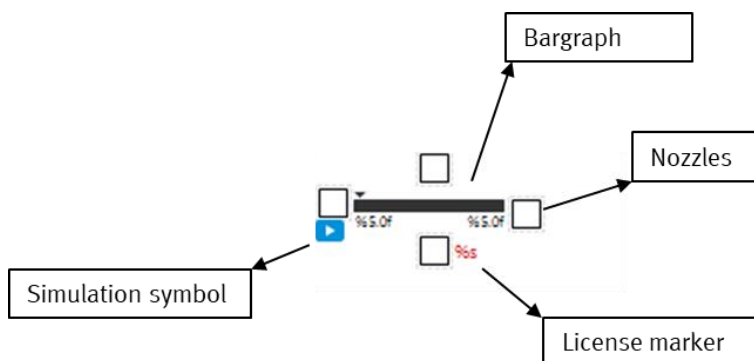


Fig. 5 Symbol hAnaViewBargraph

3.1.6.5 vAnaViewBargraph

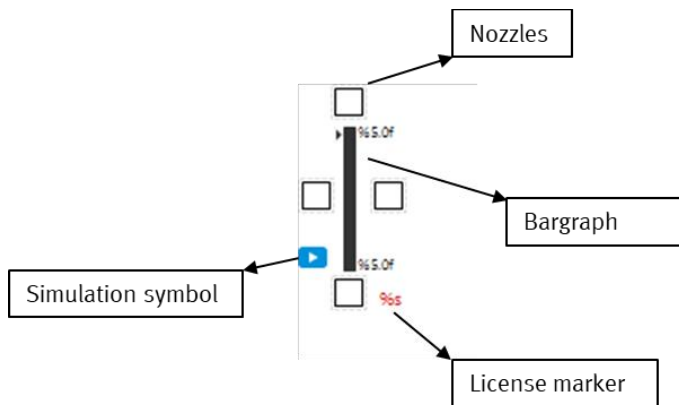


Fig. 6 Symbol vAnaViewBargraph

3.2 AnaMon function block

3.2.1 Brief description

The AnaMon function block (FB) expands the AnaView FB to include limit value monitoring.

3.2.2 Extended functions

3.2.2.1 Limit value monitoring

This function is used to monitor output value “V”. Different limit values can be entered which may not be fallen short of or exceeded by the output value.

3 monitoring categories are available to this end: Tolerance, Warning und Alarm.

There's a lower limit value for each category which may not be fallen short of (e.g., “VALLim” in the case of alarm), as well as an upper limit value which may not be exceeded (e.g., “VAHLim” in the case of alarm). The limit values are available as inputs and can be activated independent of each other (e.g., “VALEn” and “VAHEn” in the case of alarm).

Beyond this, a time can be specified via the “VALmTi” input. If a limit value violation is detected, it must persist for a duration of at least “VALmTi” in order to be recognised as an error. Otherwise, it's considered a signal flutter and no alarm is triggered.

Alarms are read out via the associated output bit (e.g., “VALAct” and “VAHAct” in the case of alarm). If a visualisation is used in CODESYS, the corresponding alarm appears in the symbol.

If limit values are set incorrectly (e.g., lower limit value > upper limit value), the WQC is set correspondingly.

3.2.2.2 Time “VALmTi”

The “VALmTi” variable is specified in seconds and is set to 0.5 s as a default value. This duration can be changed via the input at the FB or at the faceplate. Values of less than 0 are converted to positive numbers. For example, if -5 s is entered, time is set to 5 s.

3.2.2.3 Acknowledging alarms

If no limit value violations are pending anymore, the corresponding output bits are set back to FALSE. However, the error in the symbol must be reset by an operator via the faceplate (VALmAckOp) or via the “VALmAckAut” input at the FB. Visibility of the alarm symbol is described in the table.

Limit value violation still pending	Error acknowledged	Symbol
Yes	No	Blinking
Yes	Yes	Visible
No	No	Blinking
No	Yes	Invisible – no error

Tab. 8 AnaMon alarm states

Note:

- The “VALmAckAut” input is edge-triggered and only reacts to rising edges.
- When acknowledged, all currently pending alarms are acknowledged simultaneously.

- Multiple limit value violations may also be pending and can be read out via the corresponding output bits at the same time – only the highest priority is displayed in the visualisation.

3.2.3 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
				VScIMax < VScImin ⇒ VScIMax = VNormMax ⇒ VScImin = VNormMin
				VScIMax < VNormMin or VScIMax > VNormMax ⇒ VScIMax = VNormMax
				VScImin < VNormMin or VScImin > VNormMax ⇒ VScImin = VNormMin
				VScIMax – VScImin = 0 ⇒ VScIMax = VNormMax ⇒ VScImin = VNormMin
				Limit values > VNormMax and/or limit values < VNormMin (Only when they're activated)
				Upper limit value < lower limit value (Only when they're activated)
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 9 AnaMon specific worst quality code

3.2.4 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
VNormMax	REAL	100	High Scale Value of Output Value (High limit of Output Value and scaling factor for V calculation)
VNormMin	REAL	0	Low Scale Value of Output Value (Low limit of Output Value and scaling factor for V calculation)
VAHEn	BOOL	FALSE	Enables Alarm High Limit (1: enabled 0: disabled)
VAHLim	REAL	0	Limit Value for Alarm High
VWHEn	BOOL	FALSE	Enables Warning High Limit (1: enabled 0: disabled)
VWHLim	REAL	0	Limit Value for Warning High
VTHEn	BOOL	FALSE	Enables Tolerance High Limit (1: enabled 0: disabled)
VTHLim	REAL	0	Limit Value for Tolerance High
VTLEn	BOOL	FALSE	Enables Tolerance Low Limit (1: enabled 0: disabled)
VTLLim	REAL	0	Limit Value for Tolerance Low
VWLEn	BOOL	FALSE	Enables Warning Low Limit (1: enabled 0: disabled)
VWLLim	REAL	0	Limit Value for Warning Low
VALEn	BOOL	FALSE	Enables Alarm Low Limit (1: enabled 0: disabled)
VALLim	REAL	0	Limit Value for Alarm Low
VALmTi	REAL	0.5	Monitor Time to recognize limit error [s]

Tab. 10 AnaMon persistent variables

3.2.5 Symbols

3.2.5.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Alarm symbols	1 : alarm is active (top priority) 2 : warning alarm is active 3 : tolerance alarm is active	See description of function block and alarm function.
Simulation symbol	Indicates when the output value is simulated. The output value is entered via “VSim”.	Appears when VSimEn = TRUE.
Output value	Output value “V” is displayed.	Max. 5 places before and 2 places after the decimal point are displayed.
Unit	U/M “VUnit” is displayed.	Only values from the “UnitSettings” enumeration are possible.
Nozzles	Nozzles for connecting measurement lines (solid lines) and logical lines (dashed lines).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
Bargraph	Graphically represents the selected limit values and the current output value. Limit values are only taken into consideration when limit value monitoring has been activated.	Limit values must be set correctly, because the bar graph will Otherwise, be incorrectly displayed (For example, limit value High must be greater than limit value Low). “VScIMin” and “VScIMax” scale the bar graph.
Licence marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 11 AnaMon symbol description

3.2.5.2 hAnaMonDefault

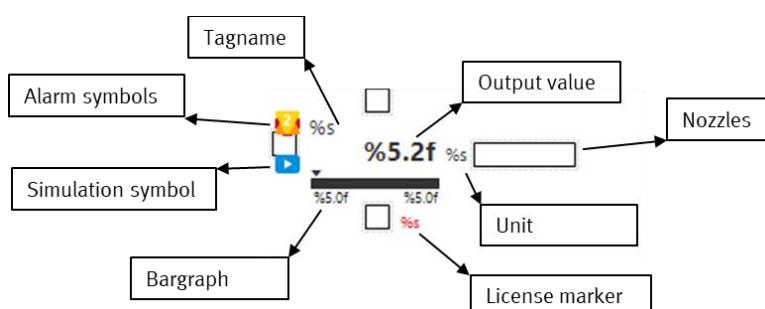


Fig. 7 Symbol hAnaMonDefault

3.2.5.3 hAnaMonValue

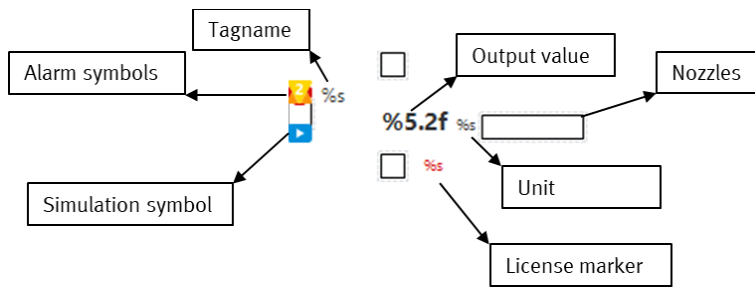


Fig. 8 Symbol hAnaMonValue

3.2.5.4 hAnaMonBargraph

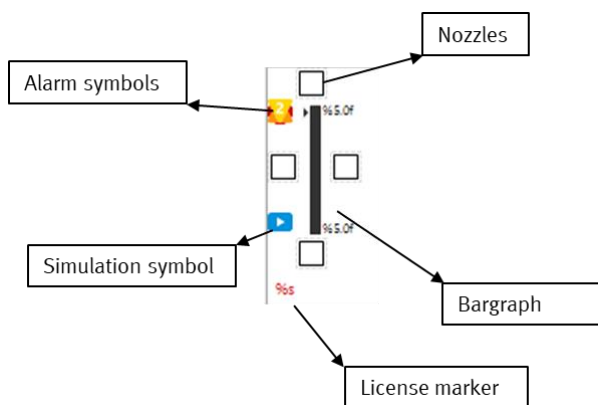


Fig. 9 Symbol hAnaMonBargraph

3.2.5.5 vAnaMonBargraph

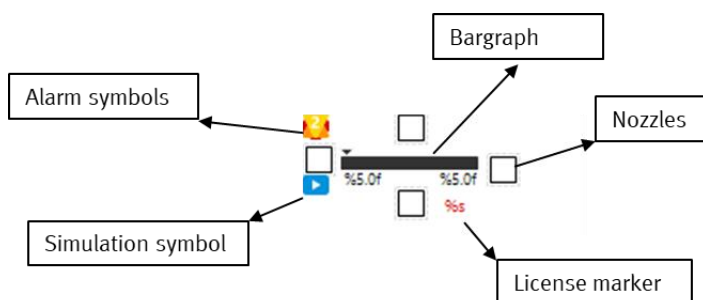


Fig. 10 Symbol vAnaMonBargraph

3.2.6 Faceplates

3.2.6.1 Main

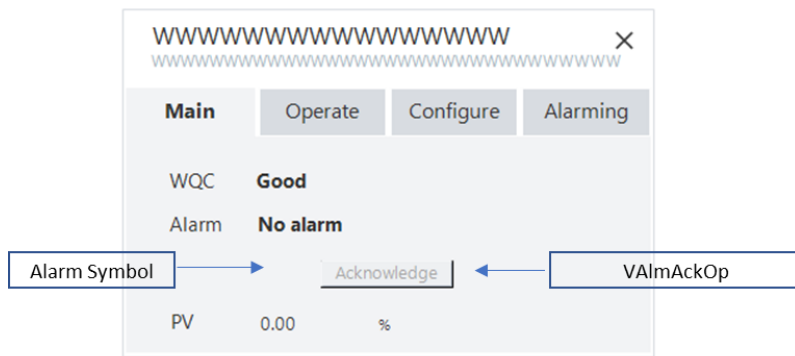


Fig. 11 AnaMon main faceplate

3.2.6.2 Operate

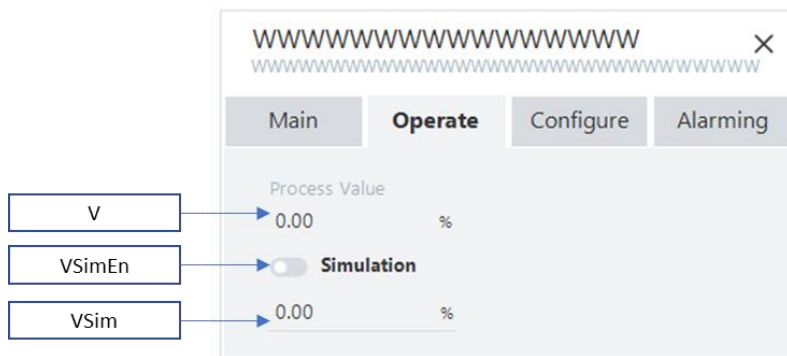


Fig. 12 AnaMon operate faceplates

3.2.6.3 Configure

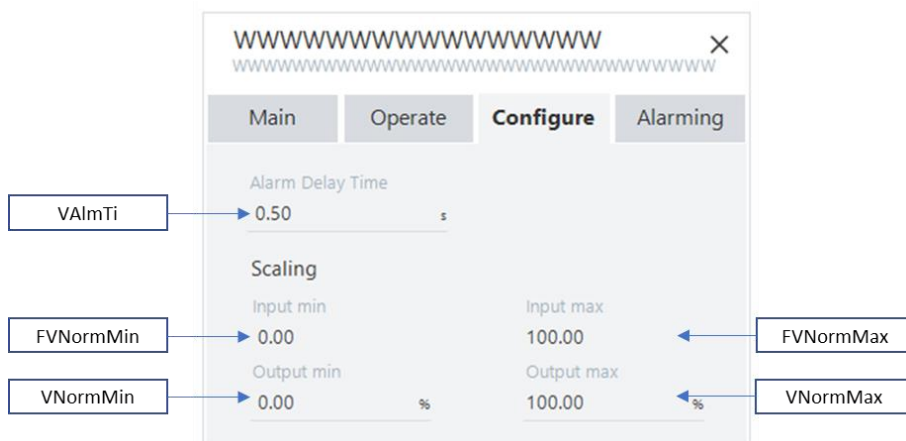


Fig. 13 AnaMon configure faceplate

3.2.6.4 Alarming

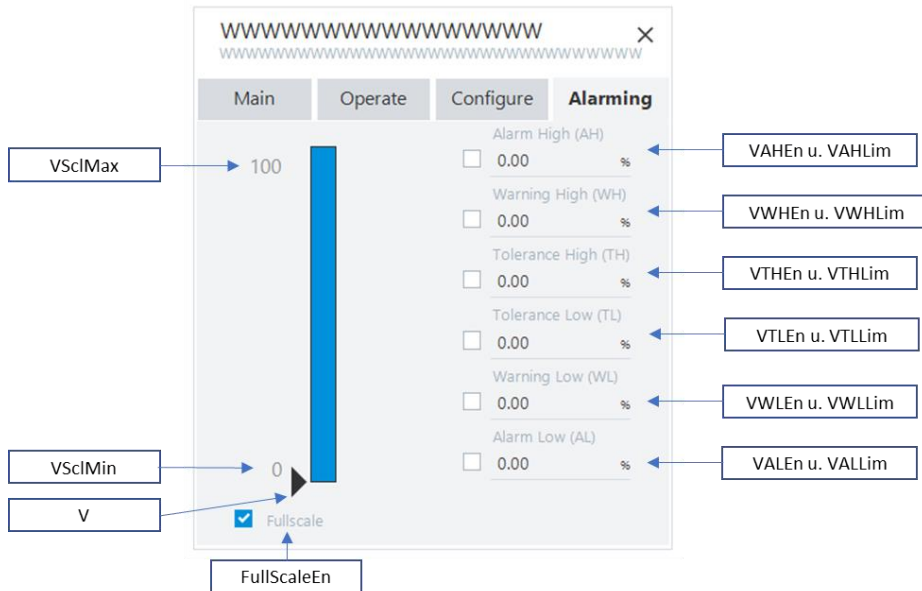


Fig. 14 AnaMon alarming faceplate

4 Analogue value operation

4.1 AnaMan function block

4.1.1 Brief description

The AnaMan function block (FB) can be used to transfer an analogue value (floating-point number) from the process field level. The signal can be further processed within the application or read out to an analogue output card.

4.1.2 Basic functions

4.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

4.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 12 Worst quality code enumeration

4.1.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g., local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

4.1.3 Functions

4.1.3.1 Input value “VMan”

The input value is represented by the “VMan” variable and is made available as an OPC UA variable. Scaling of the input value is adjusted via standardisation values “VMin” and “VMax”. Both of these values are available as inputs at the FB.

4.1.3.2 Output value “VOut”

Input value “VMan” is converted to output value “VOut” with the help of the standardisation values (“VMin”, “VMax”, “VOutMin” and “VOutMax”) and is read out from the FB. As is also the case for the standardisation values for the input value, the standardisation values for “V” are available as inputs at the FB. If “V” exceeds or falls short of its standardisation values (“VOutMin” and “VOutMax”), the corresponding standardisation value is read out as output “VOut”.

4.1.3.3 Output “VRbk”

The “VRbk” output reads out the value from “VMan”, but without limiting the value to the standardisation values.

4.1.3.4 Feedback signal “VFbk”

The “VFbkSignal” input can be linked directly to the feedback signal from the field. It’s forwarded to the “VFbk” output for further processing.

4.1.3.5 Conversion of input value “VMan” to output value “VOut”

The standardisation values, which can be read into the FB as inputs, are required for the input as well as for the output in order to convert the input value to an output value.

The conversion formula is as follows:

$$VOut = \frac{VOutMax - VOutMin}{VMax - VMin} * (VMan - VMin) + VOutMin$$

Input: VMin and Vmax

Output: VOutMin and VOutMax

4.1.3.6 Unit of measure for “VMan” and “VOut”

A unit of measure can be entered for the input value via the “VUnit” input variable at the FB. A unit of measure can also be set for the output value via the “VOutUnit” input. The “UnitSettings” enumeration is used in order to specify a unit of measure. The unit of measure is only used for displays and has no influence on the conversion.

4.1.3.7 Monitoring the standardisation values

As described above, the standardisation values limit the input and output values and are used for conversion. It must also be assured that the minimum standardisation values are not greater than the associated maximum standardisation values. Otherwise, there’s no guarantee that the conversion and the graphics in the visualisation will include sensible values.

Furthermore, the difference between “VOutMax” and “VOutMin” must not be zero, because it’s not possible to divide by zero. In this case zero is read out at output value “V”.

4.1.3.8 Scaling values for the visualisation

One Min and one Max variable (“VScMin” and “VScMax”) are used to set the display limits of each visual representation of the output value. These variables are used exclusively for the representation of graphic elements in the visualisation.

4.1.3.9 Safety position

A safety position can be entered via the “VSafeValue” input. In the event of an error (see lock functions), this value is read out from the “VOut” output. If the safety position is read out at “VOut”, the “SafePosAct” output is additionally set to TRUE.

4.1.3.10 Lock functions

4.1.3.10.1 Permit

The permit function can be activated via the “PermEn” input (= TRUE). If the “Permit” input is set to TRUE, controller commands can be processed. If the “Permit” input is set to FALSE, control commands are no longer processed. Control commands (e.g. for valves) are no longer executed. No control outputs are set either (e.g. at the operators). The FB retains the most recent state – only the safety position/value can be approached. The FB then remains in this position until the “Permit” input is set back to TRUE.

4.1.3.10.2 Interlock

The interlock function can be activated via the “IntlEn” input (TRUE). If the “Interlock” input is set to TRUE, control commands can be processed. If the “Interlock” input is set to FALSE, no control commands are processed. The safety position/value is approached, if the FB has one. If not, the most recent state is retained. No control commands are processed until the “Interlock” input is set back to TRUE or the function is deactivated via “IntlEn”. When the “Interlock” input is once again set to TRUE, the FB remains in the safety state until a new control command is transmitted.

If the FB is in the safety state, “SafePosAct” is set to TRUE.

4.1.3.10.3 Protection lock

The protection lock function can be activated via the “ProtEn” input (TRUE). If the “Protect” input is set to TRUE, control commands can be processed. If the “Protect” input is set to FALSE, no control commands are processed. The safety position/value is approached, if the FB has one. If not, the most recent state is retained. Like the interlock function, no control commands are processed until the “Protect” input is set back to TRUE or the function is deactivated via “ProtEn”. The protection lock must be additionally reset. This is possible in the visualisation via the faceplate (“ResetOp”) or via the “ResetAut” input at the FB. If the FB is in the safety state, “SafePosAct” is set to TRUE.

Note:

- The lock function is active in all states (offline, operator, automatic).
- “ResetAut” is edge-triggered and reacts to rising edges.

4.1.4 FB-specific worst quality code (WQC)

In accordance with the MTP specification, no WQC is provided for in this configuration level.

4.1.5 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
VMin	REAL	0	High Scale Value of Field Value (Scaling factor for VOut calculation)
VMax	REAL	100	Low Scale Value of Field Value (Scaling factor for VOut calculation)
SafeValue	REAL	0	Safe Value for Exception
OSLevel	Byte	0	OS Level variable

Tab. 13 AnaMan persistent variables

4.1.6 Symbols

4.1.6.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Interlock	: no interlock or protection lock is active  : interlock is active  : protection lock is active	See lock function description.
Permission	 : access permitted  : access denied	See lock function description.
Input value	“VMan” is read out.	Standardisation values are taken into consideration.
Unit	U/M “VUnit” is displayed for the “VMan” input value for both the setpoint value and the process value.	Only values from the “UnitSettings” enumeration are possible.
Nozzles	Nozzles for connecting logical lines (dashed nozzles).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
Licence marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 14 AnaMan symbol description

4.1.6.2 hAnaManValue

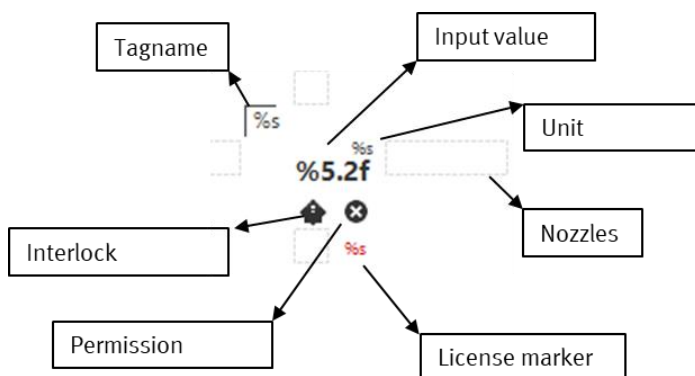


Fig. 15 Symbol hAnaManValue

4.2 AnaManInt function block

4.2.1 Brief description

The AnaManInt function block (FB) expands the AnaMan FB to include an additional input value (VInt), so that an input value can be set from two different sources.

4.2.2 Extended functions

4.2.2.1 Input values “VMan” and “VInt”

In the case of the AnaMan FB, the “VMan” input was available as an OPC UA variable. In the case of this FB, an additional “VInt” input value is available as an input at the FB.

Which input will be used for calculating “VOut” can be set via the source mode. Use of the source mode is explained in a separate section.

The selected standardisation values (“VMin” and “VMax”) are valid for both inputs, as is “VUnit”.

4.2.2.2 Output “VRbk”

Depending on the source mode setting, the “VRbk” output reads out the value from “VMan or “VInt”, but without limiting the value to the standardisation values.

4.2.3 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
40d	16#28	2	Bad, conditioned by process	VMin > VMax
				VOutMin > VOutMax
				VMax-VMin = 0 ⇒ V = 0
				V < VOutMin ⇒ V = VOutMin
				V > VOutMax ⇒ V = VOutMax
				VScIMax < VScIMin ⇒ VScIMax = VOutMax ⇒ VScIMin = VOutMin
				VScIMax < VOutMin oder VScIMax > VOutMax ⇒ VScIMax = VOutMax
				VScIMin < VOutMin oder VScIMin > VOutMax ⇒ VScIMin = VOutMin
				VScIMax - VScIMin = 0 ⇒ VScIMax = VOutMax ⇒ VScIMin = VOutMin
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 15 AnaManInt specific worst quality code

4.2.4 Source mode

4.2.4.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (manual). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (*Int) the SrcIntAct output is set to TRUE. When the manual source is active (*Man), the SrcManAct output bit is set to TRUE.

The internal source is active as a default setting.

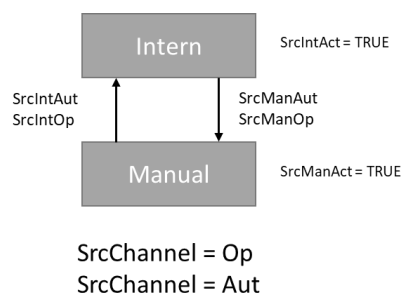


Fig. 16 States of source mode

4.2.4.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutoOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcManAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAut” and “SrcManAut” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcManAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcManAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

4.2.4.3 Modul-external access (OPC UA)

The “SrcIntOp” and “SrcManOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “SrcIntOp” (internal source) or “SrcManOp” (manual source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcManOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcManAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntOp” and “SrcManOp”) are set simultaneously, “SrcIntOp” has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “SrcIntOp”, “SrcManOp”, “SrcIntAut” or “SrcManAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

4.2.5 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It’s even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
VMin	REAL	0	High Scale Value of Field Value (Scaling factor for VOut calculation)
VMax	REAL	100	Low Scale Value of Field Value (Scaling factor for VOut calculation)
SafeValue	REAL	0	Safe Value for Exception
OSLevel	Byte	0	OS Level variable

Tab. 16 AnaManInt persistent variables

4.2.6 Symbols

4.2.6.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Interlock	: no interlock or protection lock is active  : interlock is active  : protection lock is active	See lock function description.
Permission	 : access permitted  : access denied	See lock function description.
Input value	“VMan” or “VInt” is read out.	Depending on source mode. Standardisation values are taken into consideration.
Unit	U/M “VUnit” is displayed for the “VMan” or “VInt” input value for the setpoint value and the process value.	Only values from the “UnitSettings” enumeration are possible.
Nozzles	Nozzles for connecting logical lines (dashed nozzles).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
Licence marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 17 AnaManInt symbol description

4.2.6.2 hAnaManIntValue

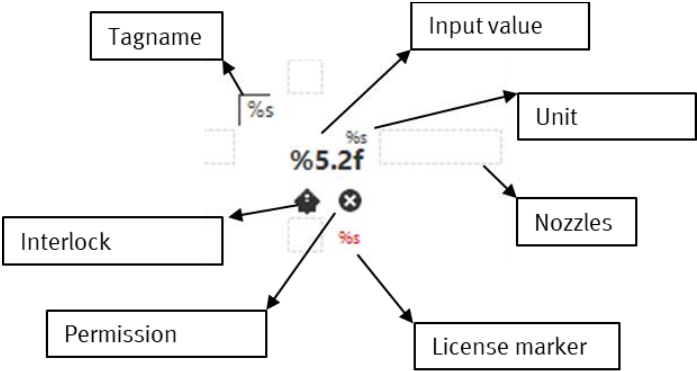


Fig. 17 Symbol hAnaManIntValue

4.2.7 Faceplates

4.2.7.1 Main

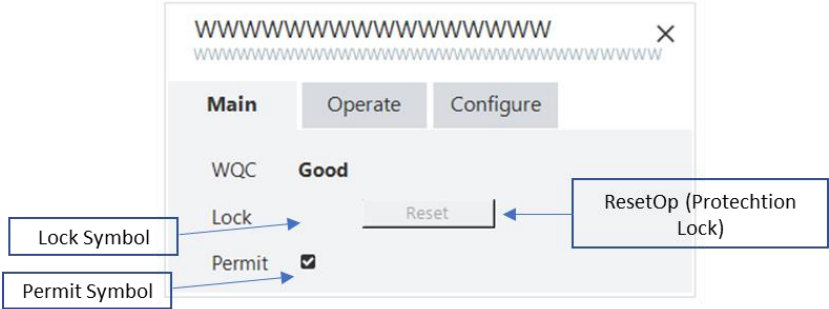


Fig. 18 AnaManInt main faceplate

4.2.7.2 Operate

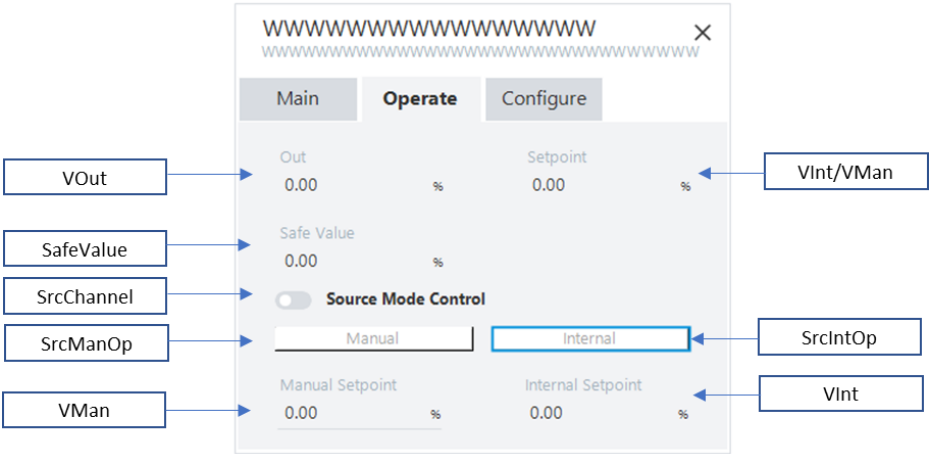


Fig. 19 AnaManInt operate faceplate

4.2.7.3 Configure

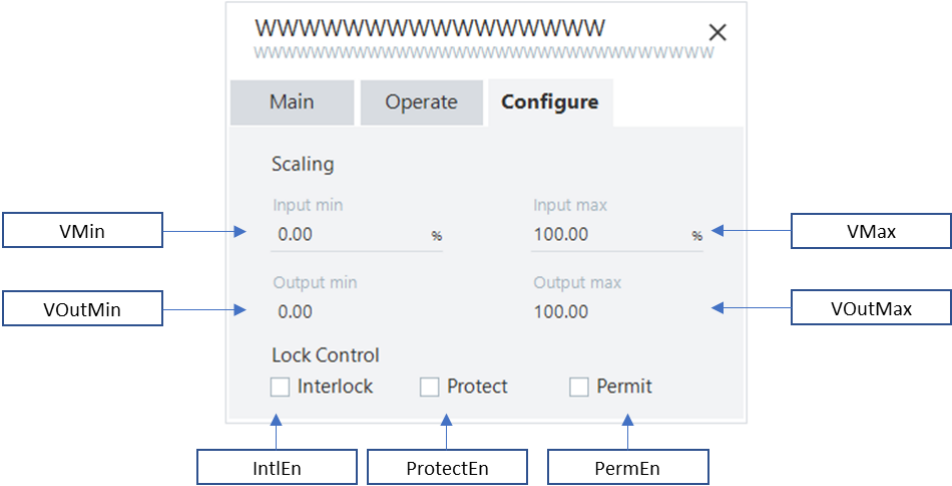


Fig. 20 AnaManInt configure faceplate

5 Digital value display

5.1 DIntView function block

The DIntView function block (FB) makes it possible to scale an analogue input value (e.g. from an analogue input card). This converted value is made available as an output for further processing. All views are processed as data type DINT.

5.1.1 Basic functions

5.1.1.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

5.1.1.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 18 Worst quality code enumeration

5.1.2 Functions

5.1.2.1 Input values "FV" and "VSim" (field value or simulated value)

The input value is represented by the "FV" variable and is made available to the FB as an input. This input can be linked to a variable or an address of an analogue input card. Scaling of the input value is adjusted via standardisation values "FVNormMin" and "FVNormMax". Both of these values are also inputs at the FB.

If no input value is available, a value can be simulated at the FB via the "VSim" input. The simulation is activated via the "VSimEn" input bit (= TRUE). Output value "V" is simulated here directly. This means that the simulated value is not scaled. It's only checked to determine if the simulated values fall short of the "VNormMin" value or exceed the "VNormMax" value. If the limit values are not complied with, the corresponding limit value is read out as "V".

5.1.2.2 Output value “V”

Input value “FV” is converted to output value “V” with the help of the standardisation values (“FVNormMin”, “FVNormMax”, “VNormMin” and “VNormMax”) and is read out from the FB. As is also the case for the standardisation values for the input value, the standardisation values for “V” are available as inputs at the FB. If “V” exceeds or falls short of its standardisation values (“VNormMin” and “VNormMax”), the corresponding standardisation value is read out as an output.

If the simulation is used, “Simulated Value” is set as the WQC. The simulated value is not scaled for read-out at output “V”. This means that only limit value monitoring takes place, but no conversion.

5.1.2.3 Conversion of input value “FV” to output value “V”

The standardisation values, which can be read into the FB as inputs, are required for the input as well as for the output in order to convert the input value to an output value.

The conversion formula is as follows:

$$V = \frac{VNormMax - VNormMin}{FVNormMax - FVNormMin} * (FV - FVNormMin) + VNormMin$$

Input: FVNormMin and FVNormMax

Output: VNormMin and VNormMax

5.1.2.4 Unit of measure for “V”

A unit of measure can be entered for the output value via the “VUnit” input variable at the FB. The “UnitSettings” enumeration is used in order to specify a unit of measure. The unit of measure is only used for displays and has no influence on the conversion.

5.1.2.5 Monitoring the standardisation values

As described above, the standardisation values limit the input and output values and are used for conversion. It must also be assured that the minimum standardisation values are not greater than the associated maximum standardisation values. Otherwise, there’s no guarantee that the conversion and the graphics in the visualisation will include sensible values.

Furthermore, the difference between “FVNormMax” and “FVNormMin” must not be zero, because it’s not possible to divide by zero. In this case zero is read out at output value “V”.

5.1.2.6 Scaling values for the visualisation

One Min and one Max variable (“VScMin” and “VScMax”) are used to set the display limits of each visual representation of the output value. These variables are used exclusively for the representation of graphic elements in the visualisation. If output “FullScEn” is set to TRUE, standardisation values “VNormMin” and “VNormMax” are used automatically as display limits. If the display limits are set incorrectly, they’re set automatically to the standardisation values. Refer to the section entitled “FB-specific worst quality code” in this regard.

5.1.3 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
				VScIMax < VScImin ⇒ VScIMax = VNormMax ⇒ VScImin = VNormMin
				VScIMax < VNormMin oder VScIMax > VNormMax ⇒ VScIMax = VNormMax
				VScImin < VNormMin oder VScImin > VNormMax ⇒ VScImin = VNormMin
				VScIMax – VScImin = 0 ⇒ VScIMax = VNormMax ⇒ VScImin = VNormMin
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 19 DIIntView specific worst quality code

5.1.4 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
VNormMax	DINT	100	High Scale Value of Output Value (High limit of Output Value and scaling factor for V calculation)
VNormMin	DINT	0	Low Scale Value of Output Value (Low limit of Output Value and scaling factor for V calculation)

Tab. 20 DIntView persistent variables

5.1.5 Symbols

5.1.5.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Simulation symbol	Indicates when the output value is simulated. The output value is entered via “VSim”.	Appears when VSimEn = TRUE.
Output value	Output value “V” is displayed.	Max. 5 places are displayed.
Unit	U/M “VUnit” is displayed.	Only values from the “UnitSettings” enumeration are possible.
Nozzles	Nozzles for connecting measurement lines (solid lines) and logical lines (dashed lines).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
Bargraph	Graphically represents the selected limit values and the current output value. Limit values are only taken into consideration when limit value monitoring has been activated.	Limit values must be set correctly, because the bar graph will Otherwise, be incorrectly displayed (For example, limit value High must be greater than limit value Low). “VScIMin” and “VScIMax” scale the bar graph.
Licence marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 21 DIntView symbol description

5.1.5.2 hDIntViewDefault

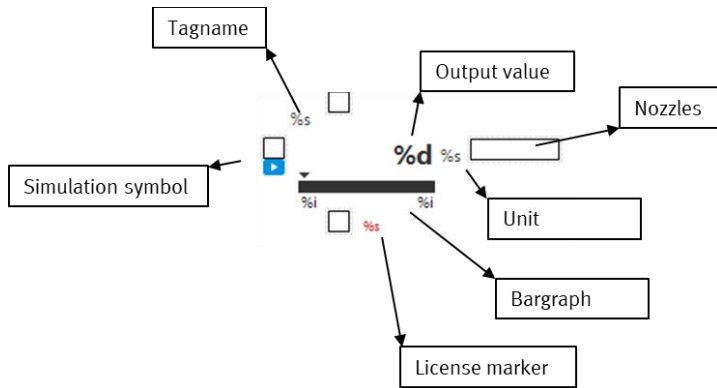


Fig. 21 Symbol hDIntViewDefault

5.1.5.3 hDIntViewValue

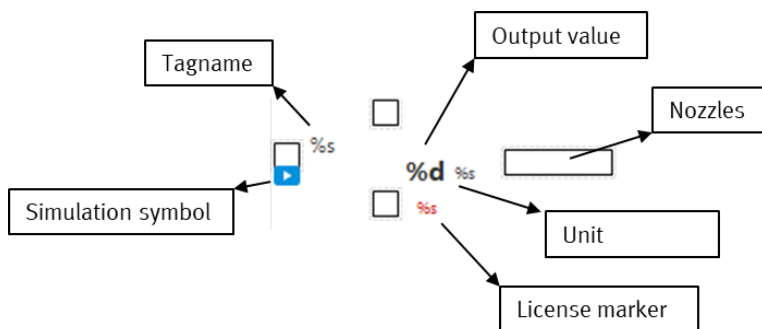


Fig. 22 Symbol hDIntViewValue

5.1.5.4 hDIntViewBargraph

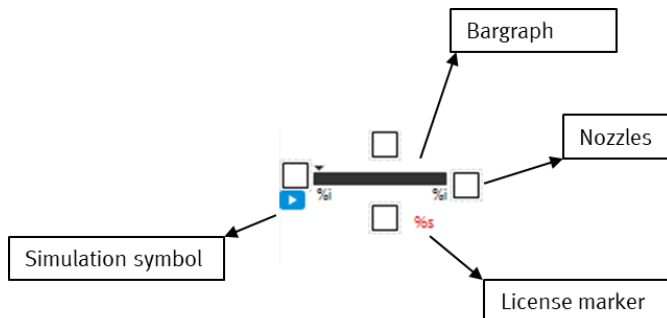


Fig. 23 Symbol hDIntViewBargraph

5.1.5.5 vDIntViewBargraph

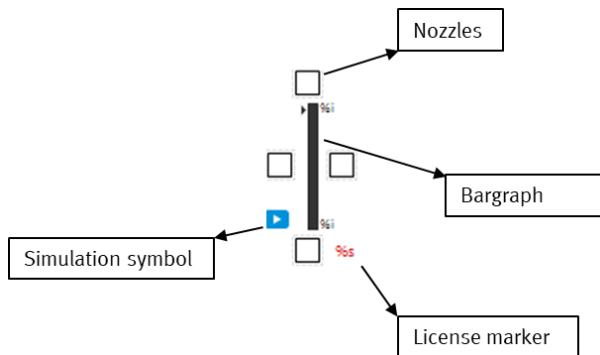


Fig. 24 Symbol vDIntViewBargraph

5.2 DIntMon function block

5.2.1 Brief description

The DIntMon function block (FB) expands the DIntView FB to include limit value monitoring.

5.2.2 Extended functions

5.2.2.1 Limit value monitoring

This function is used to monitor output value “V”. Different limit values can be entered which may not be fallen short of or exceeded by the output value.

3 monitoring categories are available to this end: Tolerance, Warning und Alarm.

There's a lower limit value for each category which may not be fallen short of (e.g. “VALLim” in the case of alarm), as well as an upper limit value which may not be exceeded (e.g. “VAHLim” in the case of alarm). The limit values are available as inputs and can be activated independent of each other (e.g. “VALEn” and “VAHEn” in the case of alarm).

Beyond this, a time can be specified via the “VALmTi” input. If a limit value violation is detected, it must persist for a duration of at least “VALmTi” in order to be recognised as an error. Otherwise, it's considered a signal flutter and no alarm is triggered.

Alarms are read out via the associated output bit (e.g. “VALAct” and “VAHAct” in the case of alarm). If a visualisation is used in CODESYS, the corresponding alarm appears in the symbol.

If limit values are set incorrectly (e.g. lower limit value > upper limit value), the WQC is set correspondingly.

5.2.2.2 Time “VALmTi”

The “VALmTi” variable is specified in seconds and is set to 0.5 s as a default value. This duration can be changed via the input at the FB or at the faceplate. Values of less than 0 are converted to positive numbers. For example, if -5 s is entered, time is set to 5 s.

5.2.2.3 Acknowledging alarms

If no limit value violations are pending anymore, the corresponding output bits are set back to FALSE. However, the error in the symbol must be reset by an operator via the faceplate (“VALmAckOp”) or via the “VALmAckAut” input at the FB. Visibility of the alarm symbol is described in the table.

Limit value violation still pending	Error acknowledged	Symbol
Yes	No	Blinking
Yes	Yes	Visible
No	No	Blinking
No	Yes	Invisible – no error

Tab. 22 Alarm symbol states

Note:

- The “VALmAckAut” input is edge-triggered and only reacts to rising edges.
- When acknowledged, all currently pending alarms are acknowledged simultaneously.

- Multiple limit value violations may also be pending and can be read out via the corresponding output bits at the same time – only the highest priority is displayed in the visualisation.

5.2.3 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
				VScIMax < VScIMin ⇒ VScIMax = VNormMax ⇒ VScIMin = VNormMin
				VScIMax < VNormMin or VScIMax > VNormMax ⇒ VScIMax = VNormMax
				VScIMin < VNormMin or VScIMin > VNormMax ⇒ VScIMin = VNormMin
				VScIMax – VScIMin = 0 ⇒ VScIMax = VNormMax ⇒ VScIMin = VNormMin
				Limit values > VNormMax and/or limit values < VNormMin (only when they're activated)
				Upper limit value < lower limit value (only when they're activated)
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 23 DIIntMon specific worst quality code

5.2.4 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
VNormMax	DINT	100	High Scale Value of Output Value (High limit of Output Value and scaling factor for V calculation)
VNormMin	DINT	0	Low Scale Value of Output Value (Low limit of Output Value and scaling factor for V calculation)
VAHEn	BOOL	FALSE	Enables Alarm High Limit (1: enabled 0: disabled)
VAHLim	DINT	0	Limit Value for Alarm High
VWHEn	BOOL	FALSE	Enables Warning High Limit (1: enabled 0: disabled)
VWHLim	DINT	0	Limit Value for Warning High
VTHEn	BOOL	FALSE	Enables Tolerance High Limit (1: enabled 0: disabled)
VTHLim	DINT	0	Limit Value for Tolerance High
VTLEn	BOOL	FALSE	Enables Tolerance Low Limit (1: enabled 0: disabled)
VTLLim	DINT	0	Limit Value for Tolerance Low
VWLEn	BOOL	FALSE	Enables Warning Low Limit (1: enabled 0: disabled)
VWLLim	DINT	0	Limit Value for Warning Low
VALEn	BOOL	FALSE	Enables Alarm Low Limit (1: enabled 0: disabled)
VALLim	DINT	0	Limit Value for Alarm Low
VALmTi	REAL	0.5	Monitor Time to recognize a limit error [s]

Tab. 24 DIntMon persistent variables

5.2.5 Symbols

5.2.5.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Alarm symbols	1 : alarm is active (top priority) 2 : warning alarm is active 3 : tolerance alarm is active	See description of function block and alarm function.
Simulation symbol	Indicates when the output value is simulated. The output value is entered via “VSim”.	Appears when VSimEn = TRUE.
Output value	Output value “V” is displayed.	Max. 5 places are displayed.
Unit	U/M “VUnit” is displayed.	Only values from the “UnitSettings” enumeration are possible.
Nozzles	Nozzles for connecting measurement lines (solid lines) and logical lines (dashed lines).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
Bargraph	Graphically represents the selected limit values and the current output value. Limit values are only taken into consideration when limit value monitoring has been activated.	Limit values must be set correctly, because the bar graph will Otherwise, be incorrectly displayed (For example, limit value High must be greater than limit value Low). “VScIMin” and “VScIMax” scale the bar graph.
Licence marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 25 DIntMon symbol description

5.2.5.2 hDIntMonDefault

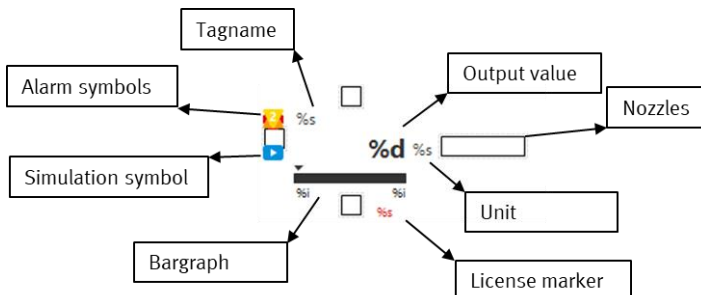


Fig. 25 Symbol hDIntMonDefault

5.2.5.3 hDIntMonValue

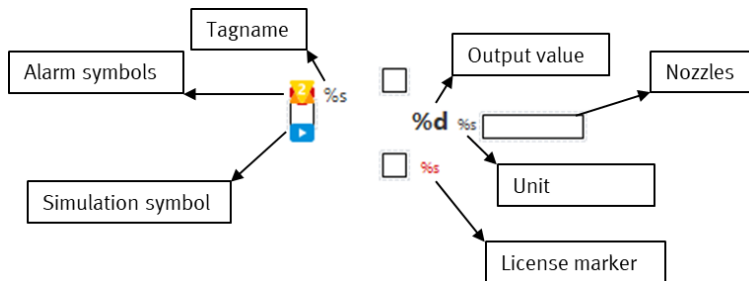


Fig. 26 Symbol hDIntMonValue

5.2.5.4 hDIntMonBargraph

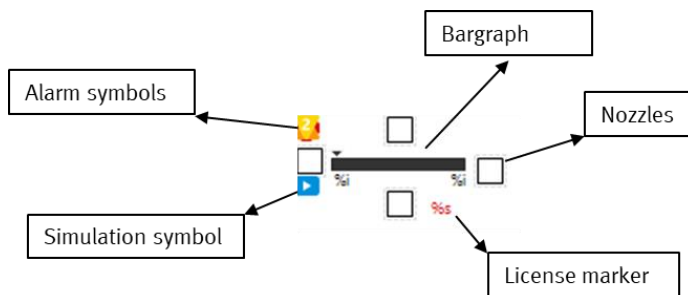


Fig. 27 Symbol hDIntMonBargraph

5.2.5.5 vDIntMonBargraph

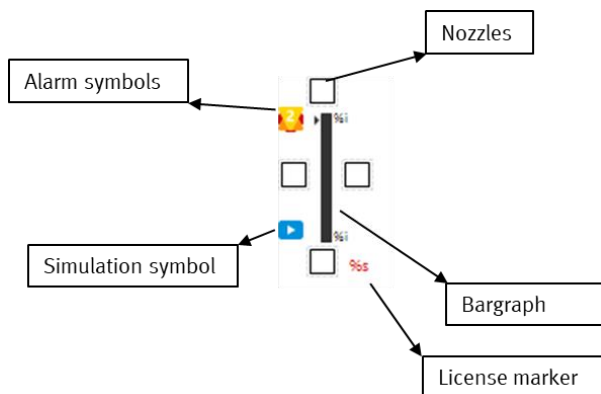


Fig. 28 Symbol vDIntMonBargraph

5.2.6 Faceplates

5.2.6.1 Main

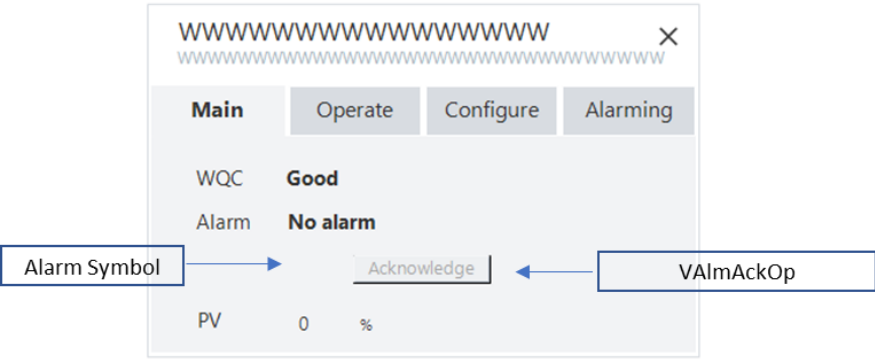


Fig. 29 DIntMon main faceplate

5.2.6.2 Operate

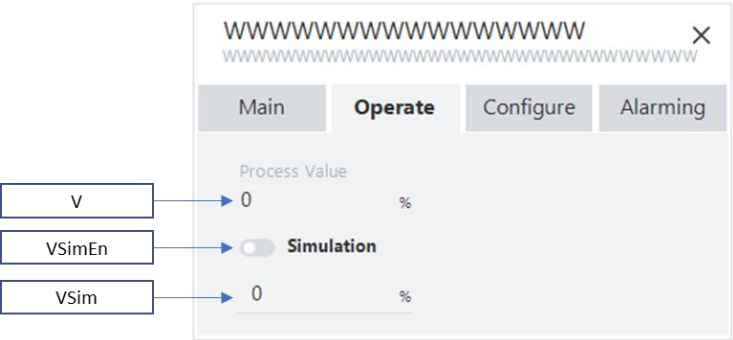


Fig. 30 DIntMon operate faceplate

5.2.6.3 Configure

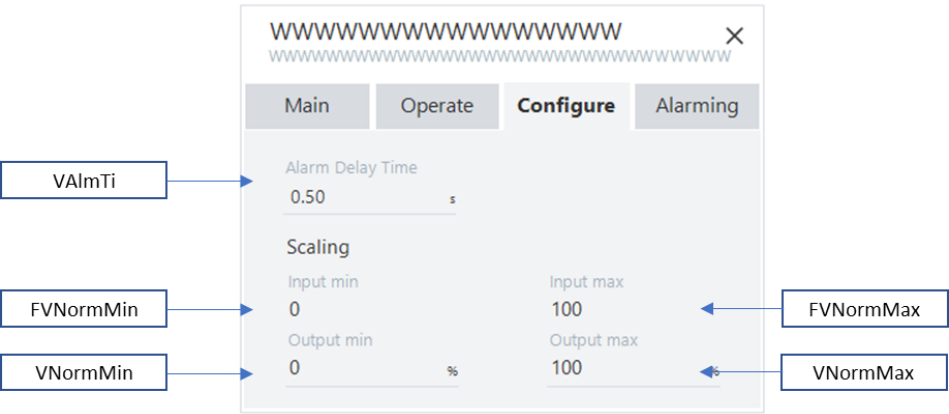


Fig. 31 DIntMon configure faceplate

5.2.6.4 Alarming

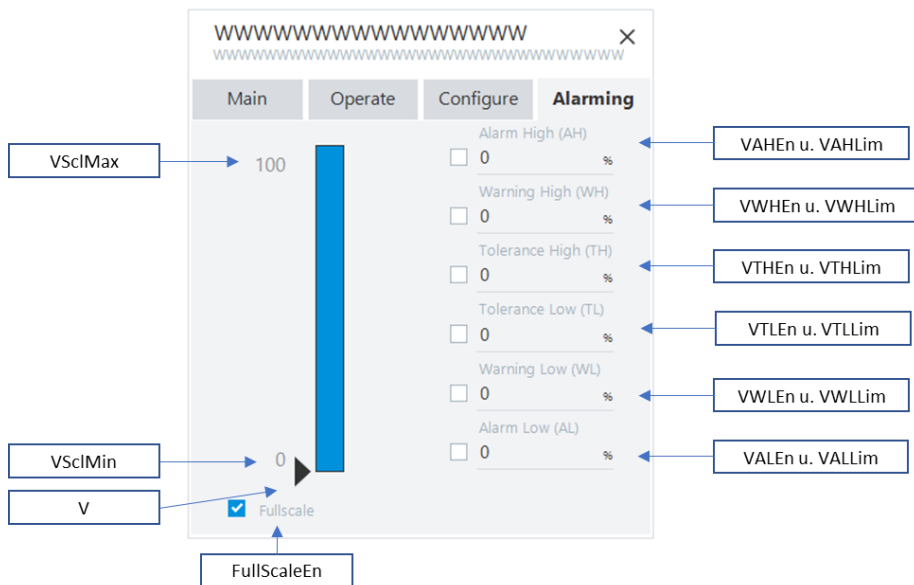


Fig. 32 DIntMon alarming faceplate

6 Digital value operation

6.1 DIntMan function block

6.1.1 Brief description

The DIntMan function block (FB) can be used to transfer an analogue value (double integer) from the process field level. The signal can be further processed within the application or read out to an analogue output card.

6.1.2 Basic functions

6.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

6.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 26 Worst quality code enumeration

6.1.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

6.1.3 Functions

6.1.3.1 Input value “VMan”

The input value is represented by the “VMan” variable and is made available as an OPC UA variable. Scaling of the input value is adjusted via standardisation values “VMin” and “VMax”. Both of these values are available as inputs at the FB.

6.1.3.2 Output value “VOut”

Input value “VMan” is converted to output value “VOut” with the help of the standardisation values (“VMin”, “VMax”, “VOutMin” and “VOutMax”) and is read out from the FB. As is also the case for the standardisation values for the input value, the standardisation values for “V” are available as inputs at the FB. If “V” exceeds or falls short of its standardisation values (“VOutMin” and “VOutMax”), the corresponding standardisation value is read out as output “VOut”.

6.1.3.3 Output “VRbk”

The “VRbk” output reads out the value from “VMan”, but without limiting the value to the standardisation values.

6.1.3.4 Feedback signal “VFbk”

The “VFbkSignal” input can be linked directly to the feedback signal from the field. It’s forwarded to the “VFbk” output for further processing.

6.1.3.5 Conversion of input value “VMan” to output value “VOut”

The standardisation values, which can be read into the FB as inputs, are required for the input as well as for the output in order to convert the input value to an output value.

The conversion formula is as follows:

$$VOut = \frac{VOutMax - VOutMin}{VMax - VMin} * (VMan - VMin) + VOutMin$$

Input: VMin and VMax

Output: VOutMin and VOutMax

6.1.3.6 Unit of measure for “VMan” and “VOut”

A unit of measure can be entered for the input value via the “VUnit” input variable at the FB. A unit of measure can also be set for the output value via the “VOutUnit” input. The “UnitSettings” enumeration is used in order to specify a unit of measure. The unit of measure is only used for displays and has no influence on the conversion.

6.1.3.7 Monitoring the standardisation values

As described above, the standardisation values limit the input and output values and are used for conversion. It must also be assured that the minimum standardisation values are not greater than the associated maximum standardisation values. Otherwise, there's no guarantee that the conversion and the graphics in the visualisation will include sensible values.

Furthermore, the difference between "VOutMax" and "VOutMin" must not be 0, because it's not possible to divide by 0. In this case zero is read out at output value "V".

6.1.3.8 Scaling values for the visualisation

One Min and one Max variable ("VScMin" and "VScMax") are used to set the display limits of each visual representation of the output value. These variables are used exclusively for the representation of graphic elements in the visualisation.

6.1.3.9 Safety position

A safety position can be entered via the "VSafeValue" input. In the event of an error (see lock functions), this value is read out from the "VOut" output. If the safety position is read out at "VOut", the "SafePosAct" output is additionally set to TRUE.

6.1.3.10 Lock functions

6.1.3.10.1 Permit

The permit function can be activated via the "PermEn" input (= TRUE). If the "Permit" input is set to TRUE, controller commands can be processed. If the "Permit" input is set to FALSE, control commands are no longer processed. Control commands (e.g. for valves) are no longer executed. No control outputs are set either (e.g. at the operators). The FB retains the most recent state – only the safety position/value can be approached. The FB then remains in this position until the "Permit" input is set back to TRUE.

6.1.3.10.2 Interlock

The interlock function can be activated via the "IntEn" input (TRUE). If the "Interlock" input is set to TRUE, control commands can be processed. If the "Interlock" input is set to FALSE, no control commands are processed. The safety position/value is approached, if the FB has one. If not, the most recent state is retained. No control commands are processed until the "Interlock" input is set back to TRUE or the function is deactivated via "IntEn". When the "Interlock" input is once again set to TRUE, the FB remains in the safety state until a new control command is transmitted. If the FB is in the safety state, "SafePosAct" is set to TRUE.

6.1.3.10.3 Protection lock

The protection lock function can be activated via the "ProtEn" input (TRUE). If the "Protect" input is set to TRUE, control commands can be processed. If the "Protect" input is set to FALSE, no control commands are processed. The safety position/value is approached, if the FB has one. If not, the most recent state is retained. Like the interlock function, no control commands are processed until the "Protect" input is set back to TRUE or the function is deactivated via "ProtEn". The protection lock must be additionally reset. This is possible in the visualisation via the faceplate ("ResetOp") or via the "ResetAut" input at the FB.

Note:

- The lock function is active in all states (offline, operator, automatic).
- "ResetAut" is edge-triggered and reacts to rising edges.

6.1.4 FB-specific worst quality code (WQC)

In accordance with the MTP specification, no WQC is provided for in this configuration level.

6.1.5 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
VMin	DINT	0	High Scale Value of Field Value (Scaling factor for VOut calculation)
VMax	DINT	100	Low Scale Value of Field Value (Scaling factor for VOut calculation)
SafeValue	DINT	0	Safe Value for Exception
OSLevel	Byte	0	OS Level variable

Tab. 27 DIntMan persistent variables

6.1.6 Symbols

6.1.6.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Interlock	: no interlock or protection lock is active  : interlock is active  : protection lock is active	See lock function description.
Permission	 : access permitted  : access denied	See lock function description.
Input value	“VMan” is read out.	Standardisation values are taken into consideration.
Unit	U/M “VUnit” is displayed for the “VMan” input value for both the setpoint value and the process value.	Only values from the “UnitSettings” enumeration are possible.
Nozzles	Nozzles for connecting logical lines (dashed nozzles).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
Licence marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 28 DIntMan symbol description

6.1.6.2 hDIntManValue

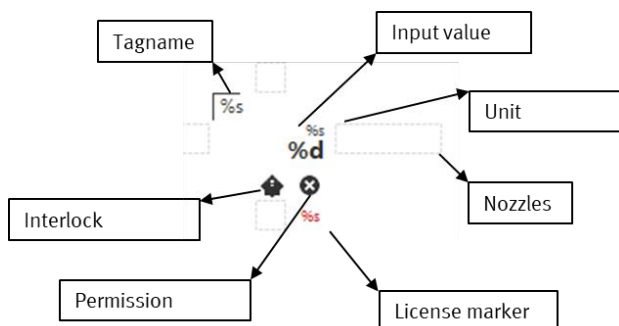


Fig. 33 Symbol hDIntManValue

6.2 DIntManInt function block

6.2.1 Brief description

The DIntManInt function block (FB) expands the DIntMan FB to include an additional input value (VInt), so that an input value can be set from two different sources.

6.2.2 Extended functions

6.2.2.1 Input values “VMan” and “VInt”

In the case of the DIntMan FB, the “VMan” input was available as an OPC UA variable. In the case of this FB, an additional “VInt” input value is available as an input at the FB.

Which input will be used for calculating “VOut” can be set via the source mode. Use of the source mode is explained in a separate section.

The selected standardisation values (“VMin” and “VMax”) are valid for both inputs, as is “VUnit”.

6.2.2.2 Output “VRbk”

Depending on the source mode setting, the “VRbk” output reads out the value from “VMan” or “VInt”, but without limiting the value to the standardisation values.

6.2.3 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
40d	16#28	2	Bad, conditioned by process	VMin > VMax
				VOutMin > VOutMax
				VMax-VMin = 0 ⇒ V = 0
				V < VOutMin ⇒ V = VOutMin
				V > VOutMax ⇒ V = VOutMax
				VScIMax < VScImin ⇒ VScIMax = VOutMax ⇒ VScImin = VOutMin
				VScIMax < VOutMin or VScIMax > VOutMax ⇒ VScIMax = VOutMax
				VScImin < VOutMin or VScImin > VOutMax ⇒ VScImin = VOutMin
				VScIMax - VScImin = 0 ⇒ VScIMax = VOutMax ⇒ VScImin = VOutMin
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 29 DINTManInt worst quality code

6.2.4 Source mode

6.2.4.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (manual). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (*Int) the SrcIntAct output is set to TRUE. When the manual source is active (*Man), the SrcManAct output bit is set to TRUE.

The internal source is active as a default setting.

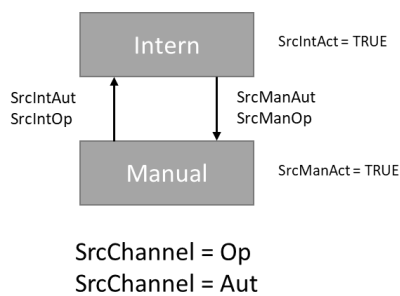


Fig. 34 Statemachine source mode

6.2.4.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutoOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcManAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAut” and “SrcManAut” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcManAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcManAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

6.2.4.3 Modul-external access (OPC UA)

The “SrcIntOp” and “SrcManOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “SrcIntOp” (internal source) or “SrcManOp” (manual source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcManOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcManAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntOp” and “SrcManOp”) are set simultaneously, “SrcIntOp” has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “SrcIntOp”, “SrcManOp”, “SrcIntAut” or “SrcManAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

6.2.5 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
VMin	DINT	0	High Scale Value of Field Value (Scaling factor for VOut calculation)
VMax	DINT	100	Low Scale Value of Field Value (Scaling factor for VOut calculation)
SafeValue	DINT	0	Safe Value for Exception
OSLevel	Byte	0	OS Level variable

Tab. 30 DIntManInt persistent variables

6.2.6 Symbols

6.2.6.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Interlock	: no interlock or protection lock is active  : interlock is active  : protection lock is active	See lock function description.
Permission	 : access permitted  : access denied	See lock function description.
Input value	“VMan” or “VInt” is read out.	Depending on source mode. Standardisation values are taken into consideration.
Unit	U/M “VUnit” is displayed for the “VMan” or “VInt” input value for the setpoint value and the process value.	Only values from the “UnitSettings” enumeration are possible.
Nozzles	Nozzles for connecting logical lines (dashed nozzles).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
Licence marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 31 DIntManInt symbol description

6.2.6.2 hDIntManIntValue

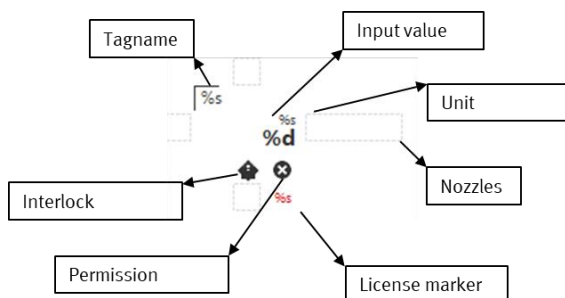


Fig. 35 Symbol hDIntManIntValue

6.2.7 Faceplates

6.2.7.1 Main

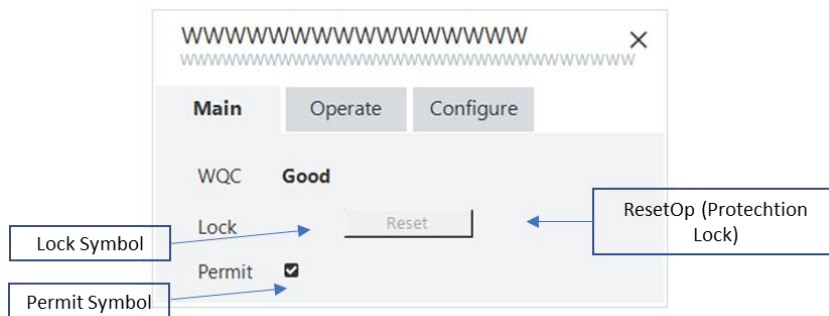


Fig. 36 DIntManInt main faceplate

6.2.7.2 Operate

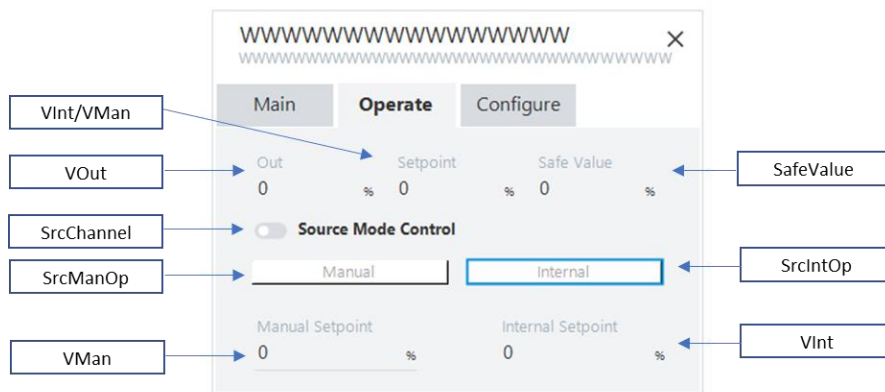


Fig. 37 DIntManInt operate faceplate

6.2.7.3 Configure

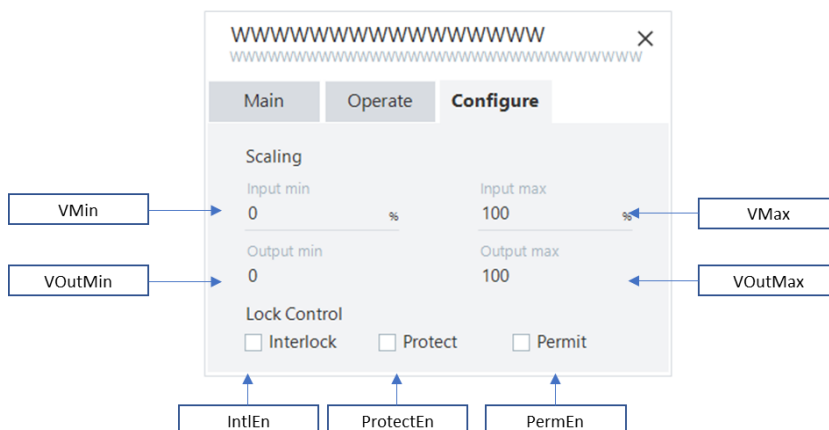


Fig. 38 DIntManInt configure faceplate

7 Binary value display

7.1 BinView function block

7.1.1 Brief description

The BinView function block (FB) makes it possible to read out a binary (digital) input value (e.g. from a binary input card) to an output for further processing.

7.1.2 Basic functions

7.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

7.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 32 Worst quality code enumeration

7.1.3 Functions

7.1.3.1 Input values "FV" and "VSim" (field value or simulated value)

The input value is represented by the "FV" variable and is made available to the FB as an input. This input can be linked to a variable or an address of a digital input card.

If no input value is available, a value can be simulated at the FB via the "VSim" input. The simulation is activated via the "VSimEn" input bit (= TRUE). Output value "V" is simulated here directly.

7.1.3.2 Status text “VState0” and “VState1”

The “State0” and “State1” inputs are available at the FB as two separate string variables. Texts can be stored for both states at these inputs. These describe the states additionally in words. For example, the text “Tank is full” can be assigned to the VState1 input for the value TRUE.

24 characters are available for each status text.

7.1.4 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 33 BinView specific worst quality code

7.1.5 Symbols

7.1.5.1 Description

Element	Description	Rule
Tagname	The name of the unit of measure is indicated in the visualisation.	Max. 16 characters.
Simulation symbol	Indicates when the output value is simulated. The output value is entered via “VSim”.	Appears when VSimEn = TRUE.
Output value	Output value “V” as text.	Depending on the value, text is displayed from “State0” or “State1”.
Nozzles	Nozzles for connecting measurement lines (solid lines) and logical lines (dashed lines).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
Licence marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 34 BinView symbol description

7.1.5.2 hBinViewValue

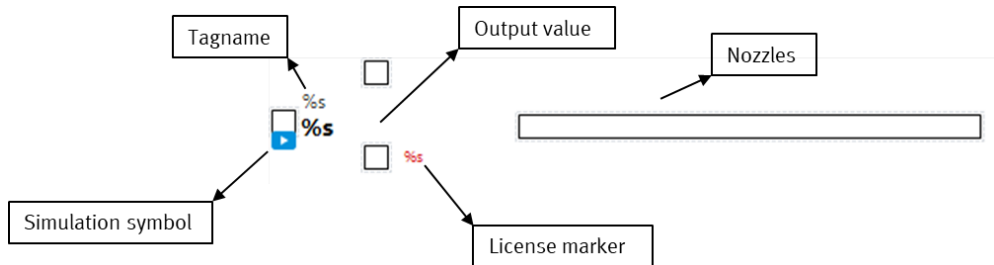


Fig. 39 Symbol hBinViewValue

7.2 BinMon function block

7.2.1 Brief description

The BinMon function block (FB) expands the BinView FB to include monitoring of an alarm state. There's additional diagnostics function as well, in order to detect signal flutter.

7.2.2 Extended functions

7.2.2.1 Diagnostics function for the detection of signal flutter

In the case of binary inputs, flutter may occur at the controller's input under certain circumstances. Flutter of this type may lead to undesirable performance of the controller application. And thus, it makes good sense to diagnose flutter in order to be aware of signal quality. Signal flutter monitoring can be switched on by setting the "VFlutEn" input at the FB to TRUE. The following inputs must be configured as well:

- VFlutTi: monitoring time during which the number of flutters is counted. The value is specified in seconds and is set to 60 s as a default value. Values of less than 0 are converted to positive numbers. For example, if -5 s is entered, time is set to 5 s.
- VFlutCnt: maximum number of fluttering signals which are permitted to occur during the time period "VFlutTi". Values of less than 0 are converted to positive numbers. For example, if -5 is entered, the quantity is set to 5.
- FlutLen: minimum duration of a flutter-free signal. If a signal is active for longer than this duration, it's considered a signal and is not deemed flutter. The value is specified in seconds and is set to 0.5 s as a default value. If values of less than 0.5 are entered, a value of 0.5 s is automatically assumed for calculation.

The result is read out at output "VFlutAct". If this output is set to FALSE, the signal is flutter-free.

7.2.2.2 Alarm state monitoring

The "V" output can assume a value of TRUE or FALSE. If an alarm is required when setting one of these two states, this function can be activated via the "VAlmStateEn" input variable (= TRUE).

The input at the "VAlmState" FB can be used to set the state for which an alarm is triggered.

In order to assure that the alarm is not triggered in the event of signal flutter, the alarm state must be pending for a certain amount of time in order to identify it as such.

A duration can be specified via the "VAlmTi" input to this end. The signal is not interpreted as an alarm state until it has been active for this duration, after which the FB reads out an error via the output by setting "VAlmStateAct" to TRUE.

7.2.2.3 Time "VAlmTi"

The "VAlmTi" variable is specified in seconds and is set to 0.5 s as a default value. This duration can be changed via the input at the FB or at the faceplate. Values of less than 0 are converted to positive numbers. For example, if -5 s is entered, time is set to 5 s.

7.2.2.4 Acknowledging alarms

When the output is no longer in the alarm state, the “VAlmStateAct” output is set back to FALSE. However, the error in the symbol must be reset by an operator via the faceplate (“VAlmAckOp”) or via the “VAlmAckAut” input at the FB. Visibility of the alarm symbol is described in the table.

Limit value violation still pending	Error acknowledged	Symbol
Yes	No	Blinking
Yes	Yes	Visible
No	No	Blinking
No	Yes	Invisible – no error

Tab. 35 Alarm symbol states

Note:

- The “VAlmAckAut” input is edge-triggered and only reacts to rising edges.

7.2.3 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 36 BinMon specific worst quality code

7.2.4 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.

- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
VFlutEn	BOOL	FALSE	Enables Fluttering Recognition (1: enabled 0: disabled)
VFlutTi	REAL	60	Period of an active signal before it will be recognized as flutter-free [s]
VFlutLen	REAL	0.5	Signal length of an active signal so that it is recognized as a signal and not as a fluttering. [s]
VFlutCnt	INT	0	Counts of the allowed fluttering signals in the defined period VFlutTi
VAlmStateEn	BOOL	FALSE	Enables the Alarm State Monitoring (1: enabled 0: disabled)
VAlmState	BOOL	FALSE	State for Alarm State Monitoring (1: TRUE is the alarm state 0: FALSE is the alarm state)
VAlmTi	REAL	0.5	Monitor Time to recognize the state as an alarm state [s]

Tab. 37 BinMon persistent variables

7.2.5 Symbols

7.2.5.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Alarm symbols	 : alarm is active (top priority)	See description of function block and alarm function.
Simulation symbol	Indicates when the output value is simulated. The output value is entered via “VSim”.	Appears when VSimEn = TRUE.
Output value	Output value “V” as text.	Depending on the value, text is displayed from “State0” or “State1”.
Nozzles	Nozzles for connecting measurement lines (solid lines) and logical lines (dashed lines).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
Licence marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 38 BinMon symbol description

7.2.5.2 hBinMonValue

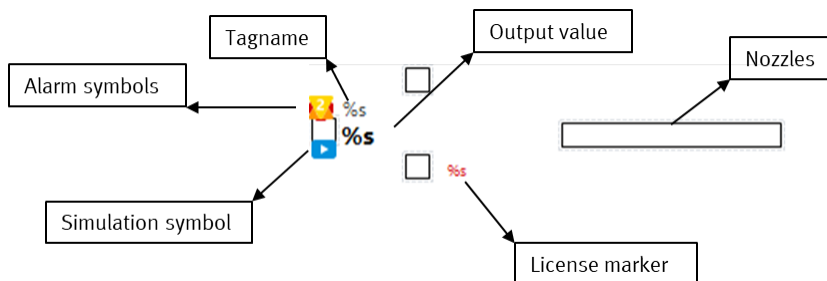


Fig. 40 Symbol hBinMonValue

7.2.6 Faceplates

7.2.6.1 Main

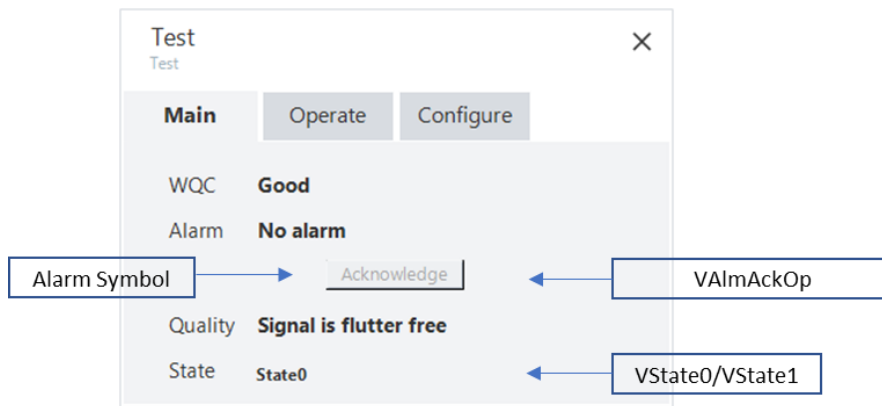


Fig. 41 BinMon main faceplate

7.2.6.2 Operate

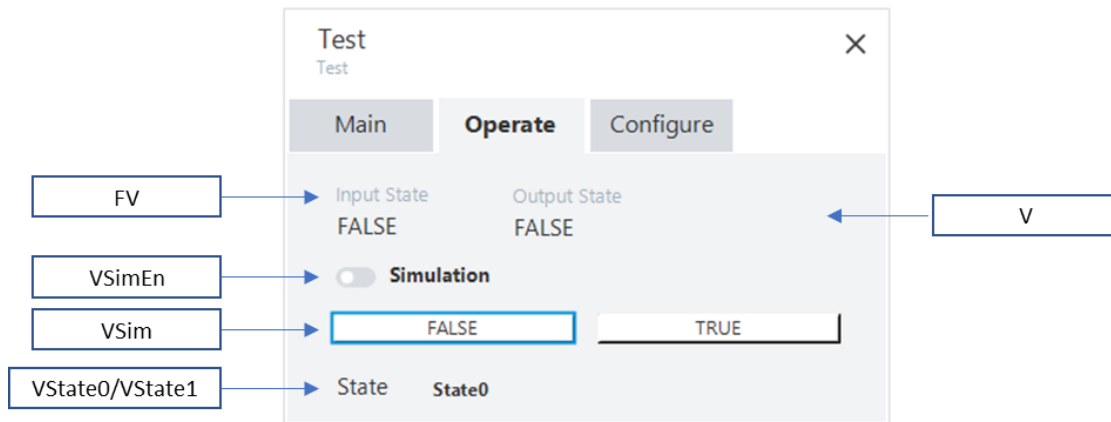


Fig. 42 BinMon operate faceplate

7.2.6.3 Configure

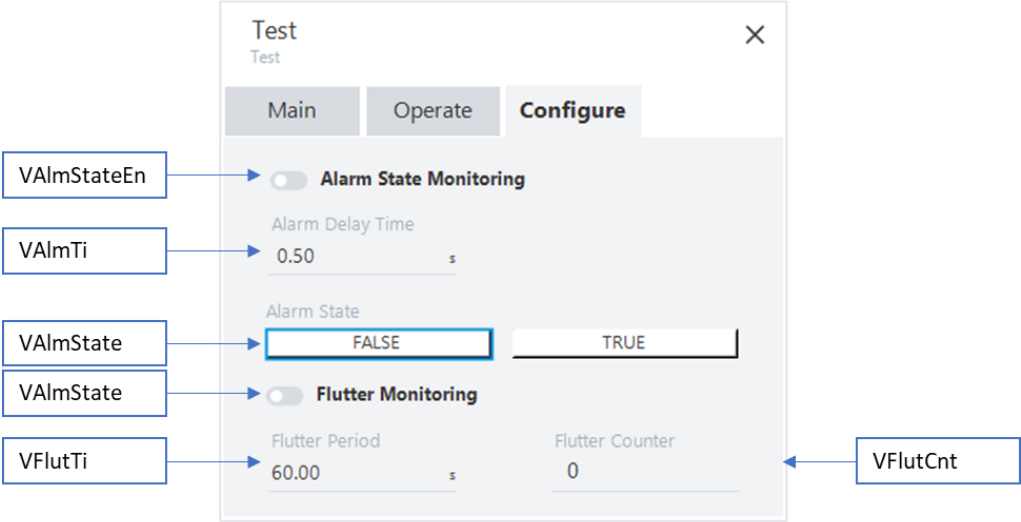


Fig. 43 BinMon configure faceplate

8 Binary operations

8.1 BinMan function block

8.1.1 Brief description

The BinMan function block (FB) can be used to transfer an binary value (TRUE/FALSE) from the process field level. The signal can be further processed within the application or read out to a digital output card.

8.1.2 Basic functions

8.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

8.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 39 Worst quality code enumeration

8.1.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

8.1.3 Functions

8.1.3.1 Input value “VMan”

The input value is represented by the “VMan” variable and is made available as an OPC UA variable. The input value is read out from the “VOut” output for further processing.

8.1.3.2 Status text “VState0” and “VState1”

The “VState0” and “VState1” inputs are available at the FB as two separate string variables. Texts can be stored for both states at these inputs. These describe the states additionally in words. For example, the text “Lamp on” can be assigned to the VState1 input for the value TRUE.

24 characters are available for each status text.

8.1.3.3 Output “VRbk”

The “VRbk” output reads out the value from “VMan” as an output.

8.1.3.4 Feedback signal “VFbk”

The “VFbkSignal” input can be linked directly to the feedback signal from the field. It's forwarded to the “VFbk” output for further processing.

8.1.3.5 Safety state

A safety position can be entered via the “VSafeState” input. In the event of an error (see lock functions), this value is read out from the “VOut” output. If the safety position is read out at “VOut”, the “SafePosAct” output is additionally set to TRUE.

8.1.3.6 Lock functions

8.1.3.6.1 Permit

The permit function can be activated via the “PermEn” input (= TRUE). If the “Permit” input is set to TRUE, controller commands can be processed. If the “Permit” input is set to FALSE, control commands are no longer processed. Control commands (e.g. for valves) are no longer executed. No control outputs are set either (e.g. at the operators). The FB retains the most recent state – only the safety position/value can be approached. The FB then remains in this position until the “Permit” input is set back to TRUE.

8.1.3.6.2 Interlock

The interlock function can be activated via the “IntlEn” input (TRUE). If the “Interlock” input is set to TRUE, control commands can be processed. If the “Interlock” input is set to FALSE, no control commands are processed. The safety position/value is approached, if the FB has one. If not, the most recent state is retained. No control commands are processed until the “Interlock” input is set back to TRUE or the function is deactivated via “IntlEn”. When the “Interlock” input is once again set to TRUE, the FB remains in the safety state until a new control command is transmitted.

If the FB is in the safety state, “SafePosAct” is set to TRUE.

8.1.3.6.3 Protection Lock

The protection lock function can be activated via the “ProtEn” input (TRUE). If the “Protect” input is set to TRUE, control commands can be processed. If the “Protect” input is set to FALSE, no control commands are processed. The safety position/value is approached, if the FB has one. If not, the most recent state is retained. Like the interlock function, no control commands are processed until the “Protect” input is set back to TRUE or the function is deactivated via “ProtEn”. The protection lock must be additionally reset. This is possible in the visualisation via the faceplate (“ResetOp”) or via the “ResetAut” input at the FB.

Note:

- The lock function is active in all states (offline, operator, automatic).
- “ResetAut” is edge-triggered and reacts to rising edges.

8.1.4 FB-specific worst quality code (WQC)

In accordance with the MTP specification, no WQC is provided for in this configuration level.

8.1.5 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It’s even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
SafeState	BOOL	FALSE	Safe State for Exception
OSLevel	BYTE	0	OS Level Variable

Tab. 40 BinMan persistent variables

8.1.6 Symbols

8.1.6.1 Description

Element	Description	Rule
Tagname	Name of the unit of measure – indicated in the visualisation.	Max. 16 characters.
Interlock	: no interlock or protection lock is active  : interlock is active  : protection lock is active	See lock function description.
Permission	 : access permitted  : access denied	See lock function description.
Input value	“VMan” is read out.	Depending on the value, text is displayed from “State0” or “State1”.
Nozzles	Nozzles for connecting logical lines (dashed nozzles).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
Licence marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 41 BinMan symbol description

8.1.6.2 hBinManValue

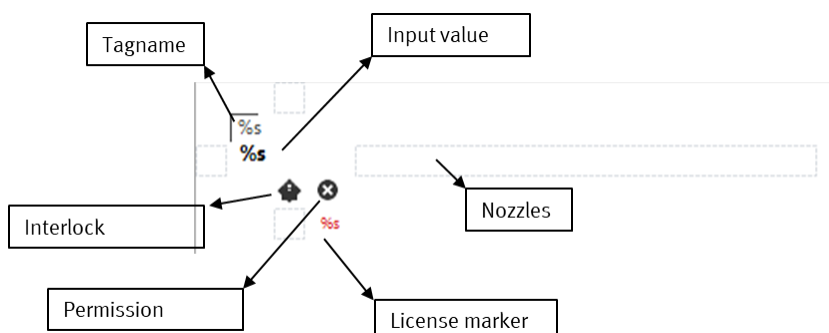


Fig. 44 Symbol hBinManValue

8.2 BinManInt function block

8.2.1 Brief description

The BinManInt function block (FB) expands the BinMan FB to include an additional input value (VInt), so that an input value can be set from two different sources.

8.2.2 Extended functions

8.2.2.1 Input values “VMan” and “VInt”

In the case of the BinMan FB, the “VMan” input was available as an OPC UA variable. With this FB there is an additional “VInt” input value as an input value at the FB.

Which input is read out to “VOut” can be set via the source mode. Use of the source mode is explained in a separate section.

8.2.2.2 Output “VRbk”

Depending on the source mode setting, the “VRbk” output reads out the value from “VMan or “VInt”.

8.2.3 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 42 BinManInt specific worst quality code

8.2.4 Source mode

8.2.4.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (manual). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (*Int) the SrcIntAct output is set to TRUE. When the manual source is active (*Man), the SrcManAct output bit is set to TRUE.

The internal source is active as a default setting.

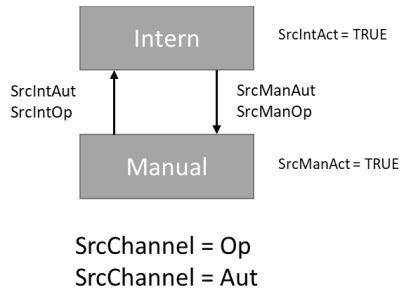


Fig. 45 Statemachine source mode

8.2.4.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcManAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAut” and “SrcManAut” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcManAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcManAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

8.2.4.3 Modul-external access (OPC UA)

The “SrcIntOp” and “SrcManOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “SrcIntOp” (internal source) or “SrcManOp” (manual source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcManOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcManAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op

variables, requests are not processed by the FB although they're acknowledged. Invalid requests are not saved for a later point in time.

If both inputs ("SrcIntOp" and "SrcManOp") are set simultaneously, "SrcIntOp" has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable "SrcIntOp", "SrcManOp", "SrcIntAut" or "SrcManAut", the variable is set back to FALSE in order to acknowledge the request. For this reason, it's important to ensure that these inputs are not permanently assigned to the FB.

8.2.5 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The "persistent variables" object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the "FestoPAToolkit" namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
SafeState	BOOL	FALSE	Safe State for Exception
OSLevel	BYTE	0	OS Level Variable

Tab. 43 BinManInt persistent variables

8.2.6 Symbols

8.2.6.1 Description

Element	Description	Rule
Tagname	Name of the unit of measure – indicated in the visualisation.	Max. 16 characters.
Interlock	: no interlock or protection lock is active  : interlock is active  : protection lock is active	See lock function description.
Permission	 : access permitted  : access denied	See lock function description.
Input value	“VMan” or “VInt” is read out.	Depending on the value, text is displayed from “State0” or “State1”.
Nozzles	Nozzles for connecting logical lines (dashed nozzles).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 44 BinManInt symbol description

8.2.6.2 hBinManIntValue

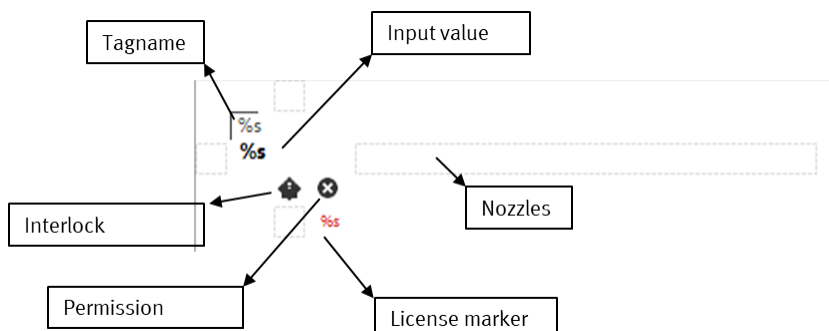


Fig. 46 Symbol hBinManIntValue

8.2.7 Faceplates

8.2.7.1 Main

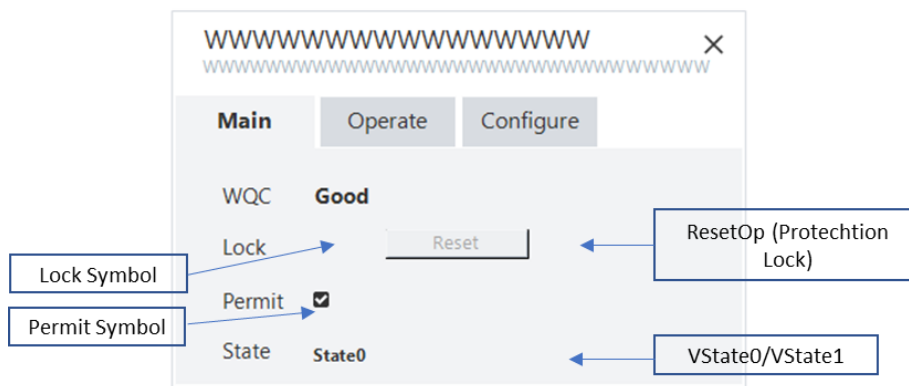


Fig. 47 BinManInt main faceplate

8.2.7.2 Operate

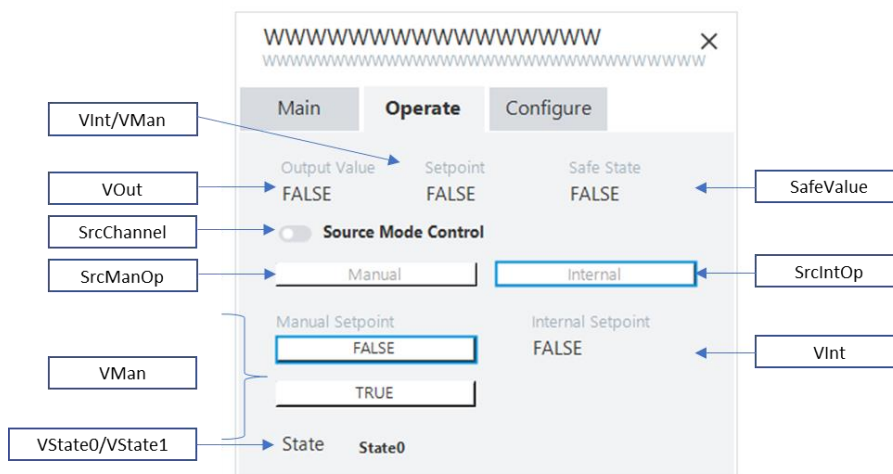


Fig. 48 BinManInt operate faceplate

8.2.7.3 Configure

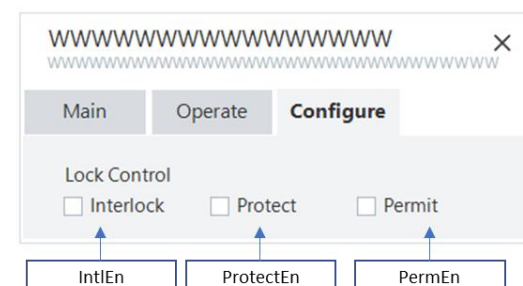


Fig. 49 BinManInt configure faceplate

9 Character string display

9.1 StringView function block

9.1.1 Brief description

The StringView function block (FB) can be used to display a string in a visualisation system.

9.1.1.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

9.1.1.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 45 Worst quality code enumeration

9.1.2 Functions

9.1.2.1 Input value "Text"

A character string can be entered via the "Text" input variable. The character string can be entered directly or via a variable. Up to 24 characters can be entered. Any additional characters are cut off in the visualisation.

9.1.3 FB-specific worst quality code (WQC)

This FB doesn't have any specific WQC.

9.1.4 Persistent variables

This FB doesn't have any persistent variables.

9.1.5 Symbols

9.1.5.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Text	Text variable is read out.	Up to 24 characters are displayed.
Nozzles	Nozzles for connecting logical lines (dashed line).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 46 StrView symbol description

9.1.5.2 hStrViewValue

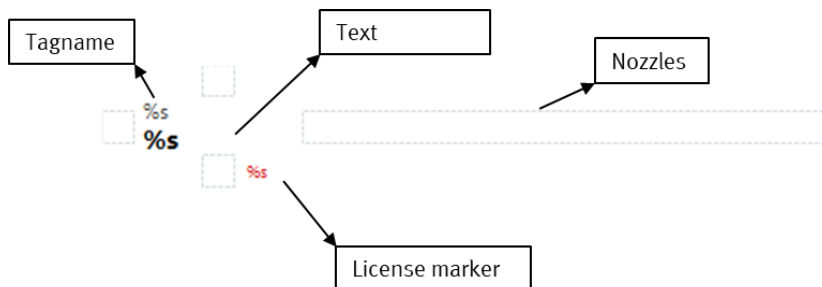


Fig. 50 Symbol hStrViewValue

9.1.6 Faceplates

There's no faceplate for this FB.

10 Controller

10.1 PIDCtrl function block

10.1.1 Brief description

The PIDCtrl function block (FB) makes information available for a continuous PID controller, which correspondingly processes the typical variables (process value PV, setpoint value SP, controlled variable MV). The interface includes a switch for the setpoint value, so that an internally specified value as well as a manually specified value can be used. The source mode function is used in this case. The controller itself has three states – offline, operator and automatic. In the operator state, controlled variable MV is specified by the operator via the “MVMan” variable. The control algorithm takes over in the automatic state. Depending on the source mode, either the internal or the manual setpoint value is processed. Limiting functions for setpoint values and controlled variables are represented with the help of the Min and Max variables. The control algorithm can be configured via the three variables P, Ti and Td.

10.1.2 Basic functions

10.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: “Tagname”.

10.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled “FB-specific worst quality code”.

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 47 Worst quality code enumeration

10.1.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

10.1.3 Functions

10.1.3.1 State machine – operation mode

The PID controller is subdivided into three states, namely offline, operator and automatic. The states are controlled via the operation mode. This mode is described in a separate section.

No values are processed in the offline state. The “MV” output is set to 0.

In the operator state, the operator can set the “MV” output to a fixed value via OPC UA or via a faceplate. This value is specified via the “MVMan” variable.

The control algorithm (PID controller) is used in the automatic state. Whether “VMan” or “VInt” will be used as a setpoint for the calculation can be set via the source mode.

10.1.3.2 Manipulated value (MVMan)

Manipulated value “MVMan” is available as an OPC UA variable or via the faceplate. This value is used in the operator state. The control algorithm is inactive in this state and the entered “MVMan” is forwarded to the MV output.

The “MVMin” and “MVMax” inputs on the FB limit “MVMan”. If “MVMan” falls short of or exceeds these limits, the respective limit value is set as output “MV”.

10.1.3.3 Process value (PV or simulated value)

The process value can be fed back to the FB via the “PV” input.

If no process value is available, a value can be simulated via the “PVSIm” input at the FB. The simulation is activated via the “PVSImEn” input bit (= TRUE).

10.1.3.4 PID controller parameters

In the automatic state, the “MV” output is calculated via the PID control algorithm. The PID controller is comprised of a proportional component (P), an integral component (Ti) and a differential component (Td). The parameters of all three values must be configured via the inputs provided for this purpose at the FB.

10.1.3.5 Setpoint specification for “SPMan” and “SPInt”

In the automatic state, an internal or a manual setpoint can be specified for the control algorithm. Internal setpoint “SPInt” is available as an input at the FB. Manual setpoint “SPMan” can be set via OPC UA or via the faceplate. Which setpoint will be used for the controller can be set via the source mode. Control of the source mode is explained in a separate section. The “SPIntMin”/“SPManMin” and “SPIntMax”/“SPManMax” inputs at the FB limit the associated setpoint. If the setpoint falls short of or exceeds its limit values, the respective limit value is set as a setpoint and is used for further processing.

10.1.3.6 Output value “MV”

Output value “MV” is specified in the operator state by means of the “MVMan” input variable. In the automatic state, “MV” is calculated by the control algorithm. The “WorkMode” influences the output as well. In the automatic mode, “MV” is not checked against limit values “MVMin” and “MVMax”.

10.1.3.7 Unit of measure for “MVMan”, “MV”, “PV”, “SPMant” and “SPInt”

Units of measure for the manipulated value, as well as for the process value, the setpoint and output value “MV”, can be entered via the inputs at the FB which are labelled Unit. The “UnitSettings” enumeration is used in order to specify a unit of measure. The units of measure are only used for displays and have no influence on the conversion.

10.1.3.8 Monitoring minimum and maximum values

As described above, the minimum and maximum values limit the input and output values. It must also be assured that the minimum limit values are not greater than the associated maximum limit values. Otherwise, there’s no guarantee that the calculations and the graphics in the visualisation will include sensible values.

10.1.3.9 Scaling values for the visualisation

One minimum and one maximum variable (“*SclMin” und “*SclMax”) are made available for setting the display limits of each visual representation of the setpoint, the output value or the process value. These variables are used exclusively for the representation of graphic elements in the visualisation. If the display limits are set incorrectly, they’re set automatically to the corresponding limit values.

10.1.3.10 “WorkMode” input parameter

The “WorkMode” input can be used to specify whether control will involve regulating a flow rate or a fill-level. If the “WorkMode” input is set to TRUE, fill-level control is involved. This means that “MV” becomes smaller in the event of a difference between the setpoint and the process value. If the “WorkMode” input is set to FALSE, flow rate control is involved and “MV” becomes a larger in the event of a difference between the setpoint and the process value.

10.1.3.11 Resetting the integral component (ResetAut/ResetOp)

If the “ResetAut” input is executed at the FB or the “ResetOp” input is executed via OPC UA or the faceplate, the integral component is reset in the PID controller. The “ResetAut” input is edge-triggered and only reacts to rising edges. If an overflow of the integral component occurs, this is indicated at the “IntegralOverflow” output.

10.1.4 Extended functions

10.1.4.1 Deviation alarm

The deviation alarm monitors the difference between the setpoint and the process value. The alarm function can be activated via the “AlmDevEn” input (=TRUE). The value of the maximum permissible difference between the setpoint and the process value can be set via the “AlmDevLim” input.

If the specified difference is exceeded, this condition must persist for a certain amount of time before an error is read out. This duration can be set via the “DevAlmTi” input value.

An alarm is read out via the associated “AlmDevAct” output bit. If a visualisation is used in CODESYS, the corresponding alarm appears in the symbol.

10.1.4.2 Time “AlmDevTi”

The “AlmDevTi” variable is specified in seconds and is set to 0.5 s as a default value. This duration can be changed via the input at the FB or at the faceplate. Values of less than 0 are converted to positive numbers. For example, if -5 s is entered, time is set to 5 s.

10.1.4.3 Acknowledging alarms

If no limit value violations are pending anymore, the “AlmDevAct” output is set back to FALSE. However, the error in the symbol must be reset by operator via the faceplate (“AlmDevAckOp”) or via the “AlmDevAckAut” input at the FB. Visibility of the alarm symbol is described in the table.

Differential violation still pending	Error acknowledged	Symbol
Yes	No	Blinking
Yes	Yes	Visible
No	No	Blinking
No	Yes	Invisible – no error

Tab. 48 Alarm symbol states

Note:

- The “AlmDevAckAut” input is edge-triggered and only reacts to rising edges.

10.1.5 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
96d	16#60	0	Simulation Value	PVSimEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	Offline mode it active
				MVMan < MVMin
				MVMan > MVMax
				SPMan < SpManMin ⇒ SPMan = SPManMin
				SPMan > SPManMax ⇒ SPMan = SPManMax
				SPInt < SPIntMin ⇒ SPInt = SPIntMin
				SPInt > SPIntMax ⇒ SPInt = SPIntMax
				SPIntMin > SPIntMax
				SPManMin > SPManMax
				MVMin > MVMax
				PVScIMax < PVScImin ⇒ PVScIMax = SPIntMax ⇒ PVScImin = SPIntMin
				SPScIMax < SPScImin ⇒ SPScIMax = SPIntMax ⇒ SPScImin = SPIntMin
				MVScIMax < MVScImin ⇒ MVScIMax = MVMax ⇒ MVScImin = MVMin
				SpScIMax < SPIntMin oder SpScIMax > SPIntMax ⇒ SPScIMax = SPIntMax
				SpScImin < SPIntMin oder SpScImin > SPIntMax ⇒ SpScImin = SPIntMin
				MVScIMax < MVMin oder MVScIMax > MVMax ⇒ MVScIMax = MVMax
				MVScImin < MVMin oder MVScImin > MVMax ⇒ MVScImin = MVMin
				PVScIMax – PVScImin = 0 ⇒ PVScIMax = SPIntMax ⇒ PVScImin = SPIntMin
				SPScIMax – SPScImin = 0 ⇒ SPScIMax = SPIntMax ⇒ SPScImin = SPIntMin
				MVScIMax – MVScImin = 0 ⇒ MVScIMax = MVMax

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
				⇒ MVScIMin = MVMin
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 49 PIDCtrl specific worst quality code

10.1.6 Source mode

10.1.6.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (manual). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (*Int) the “SrcIntAct” output is set to TRUE. When the manual source is active (*Man), the “SrcManAct” output bit is set to TRUE.

The internal source is active as a default setting.

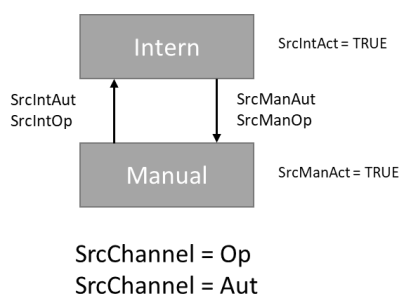


Fig. 51 Statemachine source mode

10.1.6.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutoOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcManAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAct” and “SrcManAct” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcManAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op

variables, requests are not processed by the FB although they're acknowledged. Invalid requests are not saved for a later point in time.

If both inputs ("SrcIntAut" and "SrcManAut") are set simultaneously, "SrcIntAut" has higher priority and the source is changed to or remains at internal.

10.1.6.3 Modul-external access (OPC UA)

The "SrcIntOp" and "SrcManOp" variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- "SrcChannel" must be set to "OP" via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit's faceplate).
- Set "SrcIntOp" (internal source) or "SrcManOp" (manual source) to TRUE.
- After the request, the FB sets both variables ("SrcIntOp" and "SrcManOp") to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the "SrcIntAct" and "SrcManAct" outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn't necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if "SrcChannel" is set to Op variables, requests are not processed by the FB although they're acknowledged. Invalid requests are not saved for a later point in time.

If both inputs ("SrcIntOp" and "SrcManOp") are set simultaneously, "SrcIntOp" has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable "SrcIntOp", "SrcManOp", "SrcIntAut" or "SrcManAut", the variable is set back to FALSE in order to acknowledge the request. For this reason, it's important to ensure that these inputs are not permanently assigned to the FB.

10.1.7 Operation mode

10.1.7.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode. Changes of state are only possible between offline and operator or operator and automatic. Changing directly from offline to automatic is not allowed.

The “StateChannel” input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the “StateOffAct” output is set to TRUE. When the operator mode is active “StateOpAct” is set to TRUE, and when the automatic mode is active “StateAutAct” is set to TRUE.

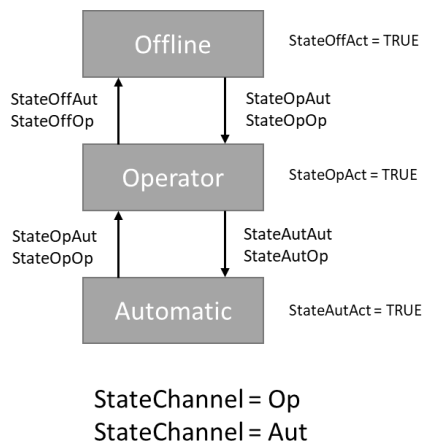


Fig. 52 Statemachine operation mode

10.1.7.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- “StateChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “StateOffAut” input (switch to offline mode), the “StateOpAut” input (switch to operator mode) or the “StateAutAut” input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch

directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

10.1.7.3 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.
- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

10.1.8 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
P	REAL	0.5	Proportional Parameter
Ti	REAL	0.5	Integration Parameter
Td	REAL	0	Derivation Parameter
WorkMode	BOOL	FALSE	Control Type Parameter (1: Level Control-MV decrease 0: Flow Control - MV increase)
AlmDevEn	BOOL	FALSE	Enables Deviation Alarm (1: enabled 0: disabled)
AlmDevLim	REAL	0	Limit Value for Deviation Alarm
AlmDevTi	REAL	0.5	Monitor Time for Deviation Alarm [s]
OSLevel	BYTE	0	OS Level Variable

Tab. 50 PIDCtrl persistent variables

10.1.9 Symbols

10.1.9.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Alarm symbols	 : alarm is active (top priority)	See description of function block and alarm function.
Operation Mode	 : offline mode is active  : operator mode is active  : automatic mode is active	See description of operation mode.
Simulation symbol	Indicates when the output value is simulated. The output value is entered via “PVSIm”.	Appears when PVSImEn = TRUE.
Process Value	“PV” or “PVSIm” is displayed.	Max. 5 places before and 2 places after the decimal point are displayed.
PV Unit	U/M “PVUnit” is displayed.	Only values from the “UnitSettings” enumeration are possible.
Setpoint	“SpInt” or “SpMan” is displayed.	Minimum and maximum values are taken into consideration.
SP Unit	U/M “SpUnit” is displayed.	Only values from the “UnitSettings” enumeration are possible.
Nozzles	Nozzles for connecting logical lines (dashed line).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
Bargraph	Graphically represents the “MV” output value. Scale “MVSclMin” and “MVSclMax”.	Scaling values must be set correctly, because the bar graph will Otherwise, be incorrectly displayed (“MVSclMin” < “MVSclMax”).
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 51 PIDCtrl symbol description

10.1.9.2 hPIDCtrlDefault

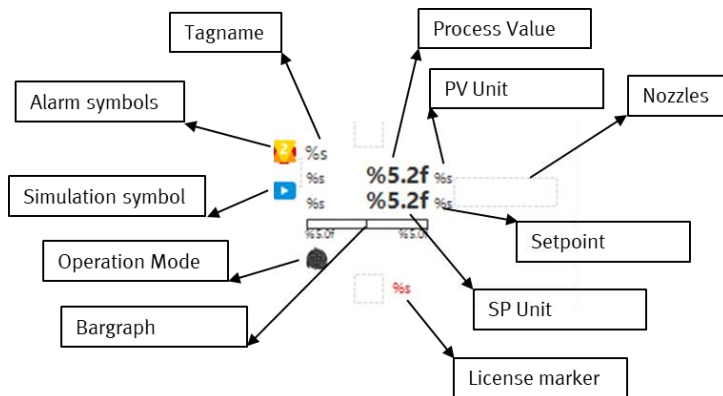


Fig. 53 Symbol hPIDCtrlDefault

10.1.10 Faceplates

10.1.10.1 Main

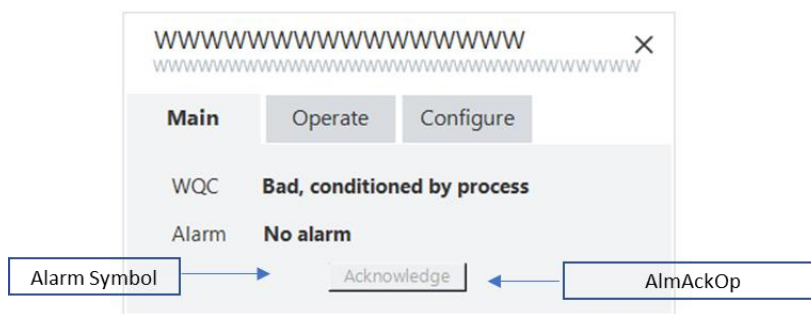


Fig. 54 PIDCtrl main faceplate

10.1.10.2 Operate

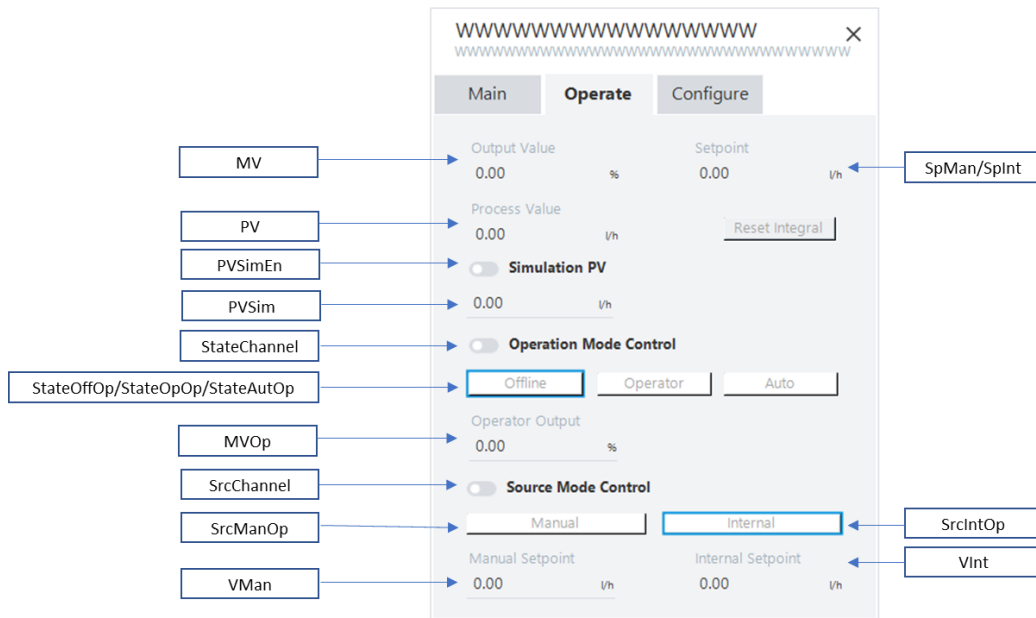


Fig. 55 PIDCtrl operate faceplate

10.1.10.3 Configure

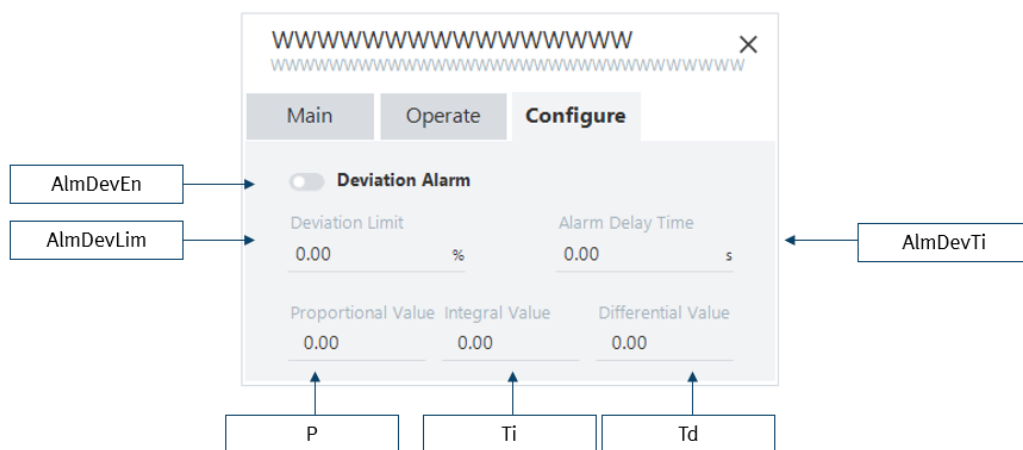


Fig. 56 PIDCtrl configure faceplate

11 Bistable valves

11.1 BinVlv function block

11.1.1 Brief description

The BinVlv function block (FB) provides the option of controlling a bistable valve (single or double-acting). Control commands can be set either directly at the FB (Aut variables) or via a faceplate / OPC UA (Op variables). Feedback signals are also available which indicate the valve's actual state.

11.1.2 Basic functions

11.1.2.1 Identification

Für jeden FB steht mindestens ein Symbol für die Visualisierung zur Verfügung. Damit das Symbol in der Visualisierung der Hardwareeinheit zugeordnet werden kann, kann direkt beim Anlegen eines Symbols ein Tag-Name und eine Tag-Description angegeben werden. Es dürfen maximal 16 Zeichen für den Tag-Name und 32 Zeichen für die Tag-Description verwendet werden, andernfalls werden die restlichen Zeichen abgeschnitten. Da diese Werte auch für einen MTP-Export relevant sind und in diesem statisch verarbeitet werden, müssen diese beiden Werte in folgendem Format angegeben werden: 'Tagname'.

11.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 52 Worst quality code enumeration

11.1.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

11.1.3 Functions

11.1.3.1 Valve control

The bistable valve can be either opened or closed. Inputs are available at the FB (OpenAut and CloseAut) and variables are available via OPC UA / the faceplate (OpenOp und CloseOp) for controlling the valve. The corresponding control command is processed by the FB and read out to the “Ctrl” output. In order to assure that double-acting valves, as well as normally closed and normally open valves can be controlled, the inverted signal is read out to “Ctrl” via the “CtrlInv” output.

Depending on the command, control commands are issued by setting the above-mentioned variables to TRUE. Input variables “OpenAut” and “CloseAut” are edge-triggered and react to rising edges. “OpenOp” and “CloseOp” are set back to FALSE after registration by the FB.

The following table shows how the outputs perform depending on the safety position and the control commands.

Control command	Output
Open	Ctrl = 1 CtrlInv = 0
Close	Ctrl = 0 CtrlInv = 1
Open and close at the same time	Ctrl = SafePos CtrlInv = !SafePos

Tab. 53 Valve control commands

The FB can be operated in three states, namely offline, operator and automatic. The states are controlled via the operation mode. This mode is described in a separate section.

The following control commands and outputs are processed and set in the respective state:

- Offline mode: Ctrl = SafePos, CtrlInv = !SafePos
- Operator mode: OpenOp and CloseOp control Ctrl and CtrlInv (can be changed via faceplate)
- Automatic mode: OpenAut and CloseAut control Ctrl und CtrlInv (inputs at the FB)

11.1.3.2 Safety position “SafePos” and “SafePosEn“

The “SafePos” input can be used to specify whether the idle or safety position is open (SafePos = TRUE) or closed (SafePos = FALSE).

The “SafePosEn” can be used to specify whether or not the valve has a safety state. If there isn’t any safety state, the most recent state is always retained in the event of an error.

11.1.3.3 Feedback signals (simulated, actual valve state)

Feedback signals are simulated automatically by the FB itself, when the “OpenFbkCalc” and/or “CloseFbkCalc” inputs are set to TRUE.

In the event that no simulation is desired, the “OpenFbkSignal” and/or “CloseFbkSignal” inputs can be linked to the valve’s actual feedback signals. Furthermore, “OpenFbkCalc” and “CloseFbkCalc” must be set to FALSE. When an end position is reached, the corresponding output at the FB (OpenFbk or CloseFbk) is set to TRUE. As a default setting, the correct feedback signals are processed via “OpenFbkSignal” and/or “CloseFbkSignal”.

11.1.3.4 Lock functions

11.1.3.4.1 Permit

The permit function can be activated via the “PermEn” input (= TRUE). If the “Permit” input is set to TRUE, controller commands can be processed. IF the “Permit” input is set to FALSE, control commands are no longer processed. Control commands (e.g. for valves) are no longer executed. No control outputs are set either (e.g. at the operators). The FB retains the most recent state – only the safety position/value can be approached. The FB then remains in this position until the “Permit” input is set back to TRUE.

11.1.3.4.2 Interlock

The interlock function can be activated via the “IntlEn” input (TRUE). If the “Interlock” input is set to TRUE, control commands can be processed. IF the “Interlock” input is set to FALSE, no control commands are processed. The safety position/value is approached, if the FB has one. If not, the most recent state is retained. No control commands are processed until the “Interlock” input is set back to TRUE or the function is deactivated via “IntlEn”. When the “Interlock” input is once again set to TRUE, the FB remains in the safety state until a new control command is transmitted.

If the FB is in the safety state, “SafePosAct” is set to TRUE.

11.1.3.4.3 Protection lock

The protection lock function can be activated via the “ProtEn” input (TRUE). If the “Protect” input is set to TRUE, control commands can be processed. IF the “Protect” input is set to FALSE, no control commands are processed. The safety position/value is approached, if the FB has one. If not, the most recent state is retained. Like the interlock function, no control commands are processed until the “Protect” input is set back to TRUE or the function is deactivated via “ProtEn”. The protection lock must be additionally reset. This is possible in the visualisation via the faceplate (“ResetOp”) or via the “ResetAut” input at the FB.

Note:

- The lock function is active in all states (offline, operator, automatic).
- “ResetAut” is edge-triggered and reacts to rising edges.

11.1.4 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
40d	16#28	2	Bad, conditioned by process	Offline mode it active
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 54 BinVlv specific worst quality code

11.1.5 Operation mode

11.1.5.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode. Changes of state are only possible between offline and operator or operator and automatic. Changing directly from offline to automatic is not allowed.

The “StateChannel” input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the “StateOffAct” output is set to TRUE. When the operator mode is active “StateOpAct” is set to TRUE, and when the automatic mode is active “StateAutAct” is set to TRUE.

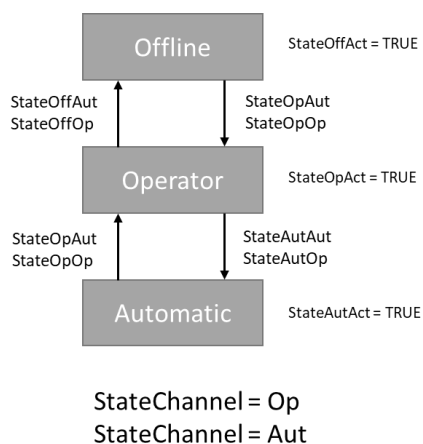


Fig. 57 Statemachine operation mode

11.1.5.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- “StateChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “StateOffAut” input (switch to offline mode), the “StateOpAut” input (switch to operator mode) or the “StateAutAut” input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.
-

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

11.1.5.3 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.
- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the

variable is set back to FALSE in order to acknowledge the request. For this reason, it's important to ensure that these inputs are not permanently assigned to the FB.

11.1.6 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
SafePosEn	BOOL	FALSE	Enables Safe Position (1: Device has a safe position 0: Device has no safe position)
SafePos	BOOL	TRUE	Safe Position (1: Open 0: Close)
OSLevel	BYTE	0	OS Level variable

Tab. 55 BinVlv persistent variables

11.1.7 Symbols

11.1.7.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Valve symbol	Grey: “CloseFbk” is active Blue: “OpenFbk” is active	
Operation Mode	 : offline mode is active  : operator mode is active  : automatic mode is active	See description of operation mode.
Interlock	 : no interlock or protection lock is active  : interlock is active  : protection lock is active	See lock function description.
Permission	 : access permitted  : access denied	See lock function description.
Nozzles	Nozzles for connecting logical lines (dashed nozzles) and pipes (blue nozzles).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 56 BinVlv symbol description

11.1.7.2 hBinVlvValue

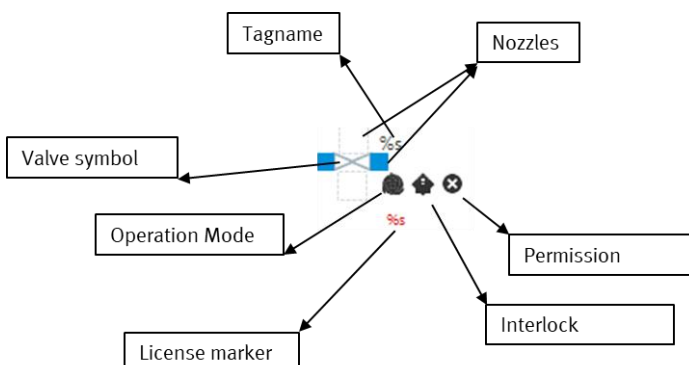


Fig. 58 Symbol hBinVlvValue

11.1.7.3 vBinVlvValue

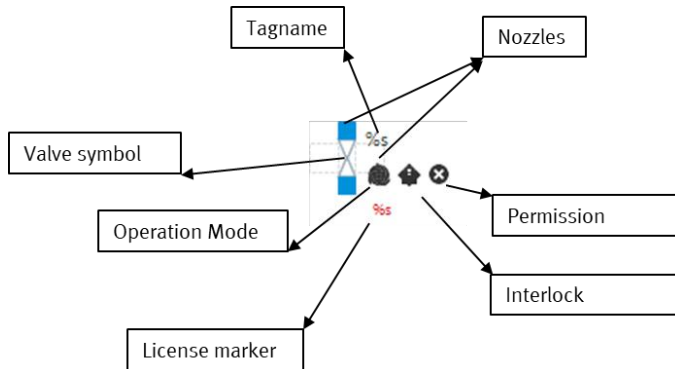


Fig. 59 Symbol vBinVlvValue

11.1.7.4 dldrThreeWayVlvValue, drdlThreeWayVlvValue, ldluThreeWayVlvValue, luldThreeWayVlvValue, rdruThreeWayVlvValue, rurdThreeWayVlvValue, ulurThreeWayVlvValue, urulThreeWayVlvValue

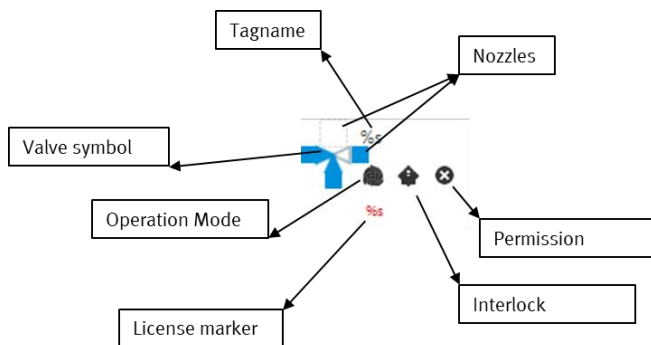


Fig. 60 Symbols Three-Way-Valves

11.2 MonBinVlv function block

11.2.1 Brief description

The MonBinVlv function block (FB) expands the BinVlv FB to include feedback monitoring for dynamic and static errors.

11.2.2 Extended functions

11.2.2.1 Activate feedback monitoring

Feedback monitoring (static and dynamic) can be activated via the “MonEn” input at the FB (= TRUE). If “MonEn” is set to FALSE or if it’s set to FALSE during operation, all error outputs are reset. The alarm symbol also disappears from the visualisation and the error doesn’t need to be acknowledged.

11.2.2.2 “MonSafePos” input

The “MonSafePos” input at the FB can be used to specify whether the valve advances to the safety position in the event of an error (MonSafePos = TRUE) or the current state is retained (MonSafePos = FALSE). In either case, the SafePosAct output bit is set to TRUE in order to detect the exception state. As a default setting, travel to the safety position takes place in the event of an error.

If the most recent state should be retained after an error, or if the valve is already in the safety state, errors are not eliminated until corresponding feedback signals provide the correct values.

If the valve is set to the safety state after an error, the error is not eliminated until the valve returns to the state in which the error occurred and the feedback signals provide the correct values.

Example:

The valve is in the closed state and needs to be opened. After opening, “OpenFbk” is still set to FALSE. The valve is thus set to its safety state again (in this case closed). In order to eliminate the error, the valve must be opened again and “OpenFbk” must be set to TRUE. Only then are the error outputs (“StatErrAct” and “DynErrAct”) set back to FALSE.

11.2.2.3 Static errors

A static error has occurred when the feedback signal changes although no change has taken place at the controller.

The following static errors are possible:

- “OpenFbk” and “CloseFbk” are both TRUE
- State is open, “OpenFbk” is set to FALSE
- State is closed, “CloseFbk” is set to FALSE

In order to assure that a malfunction is not assessed as a lost or incorrect signal, the error state must persist for a certain duration in order for it to be evaluated as an error. This duration can be set via the “MonStatTi” input at the FB. The error is not recognised and the “StatErrAct” output is not set to TRUE unless the error state is still pending after this time has elapsed.

If a visualisation is used in CODESYS, the corresponding alarm appears in the symbol.

11.2.2.4 Dynamic errors

A dynamic error has occurred when the feedback signal doesn't change although a change has taken place at the controller.

The following dynamic errors are possible:

- State should change from open to closed – “CloseFbk” remains FALSE
- State should change from closed to open – “OpenFbk” remains FALSE

In order to assure that the feedback signal is not expected immediately (because a valve requires a certain amount of time to close and open), a duration can be set via the “MonDynTi” input at the FB. If the correct feedback signal is not detected within this period of time, the “DynErrAct” output is set to TRUE.

If a visualisation is used in CODESYS, the corresponding alarm appears in the symbol.

11.2.2.5 Alarm priority

The following priorities are adhered to for the alarm display in the visualisation:

- Failed to open (top priority)
- Failed to close
- Both feedbacks active
- Open feedback lost
- Closed feedback lost
- No alarm

11.2.2.6 Durations “MonStatTi” and “MonDynTi”

The “MonStatTi” and “MonDynTi” variables are specified in seconds and are set to 0.5 s as a default value. This duration can be changed via the input at the FB or at the faceplate. Values of less than 0 are converted to positive numbers. For example, if -5 s is entered, time is set to 5 s.

11.2.2.7 Acknowledging alarms

If no errors are pending anymore, the corresponding output bits are set back to FALSE. However, the error in the symbol must be reset by operator via the faceplate (“MonAckOp”) or via the “MonAckAut” input at the FB. Visibility of the alarm symbol is described in the table.

Limit value violation still pending	Error acknowledged	Symbol
Yes	No	Blinking
Yes	Yes	Visible
No	No	Blinking
No	Yes	Invisible – no error

Tab. 57 Alarm symbol states

Note:

- The “MonAckAut” input is edge-triggered and only reacts to rising edges.
- When acknowledged, all currently pending alarms are acknowledged simultaneously.
- Multiple limit errors may also be pending – only the highest priority is displayed in the visualisation.

11.2.3 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
40d	16#28	2	Bad, conditioned by process	Offline mode it active
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 58 MonBinVlv specific worst quality code

11.2.4 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
SafePos	BOOL	FALSE	Enables Safe Position (1: Device has a safe position 0: Device has no safe position)
SafePosEn	BOOL	TRUE	Safe Position (1: Open 0: Close)
OSLevel	BYTE	0	OS Level variable
MonEn	BOOL	FALSE	Enables dynamic and static Feedback Monitoring (1: enabled 0: disabled)
MonSafePos	BOOL	TRUE	Enables Monitor Safe Position (1: go to Safe Position 0: Hold last state)

Variable	Type	Default	Description
MonStatTi	REAL	0.5	Monitor Time for uncontrolled changes [s], only positive values
MonDynTi	REAL	0.5	Monitor Time for controlled changes [s], only positive values

Tab. 59 MonBinVlv persistent variables

11.2.5 Symbols

11.2.5.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Valve symbol	Grey: “CloseFbk” is active Blue: “OpenFbk” is active	
Alarm symbols	 : alarm is active	See description of function block and alarm function.
Operation Mode	 : offline mode is active  : operator mode is active  : automatic mode is active	See description of operation mode.
Interlock	 : no interlock or protection lock is active  : interlock is active  : protection lock is active	See lock function description.
Permission	 : access permitted  : access denied	See lock function description.
Nozzles	Nozzles for connecting logical lines (dashed nozzles) and pipes (blue nozzles).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 60 MonBinVlv symbol description

11.2.5.2 hMonBinVlvValue

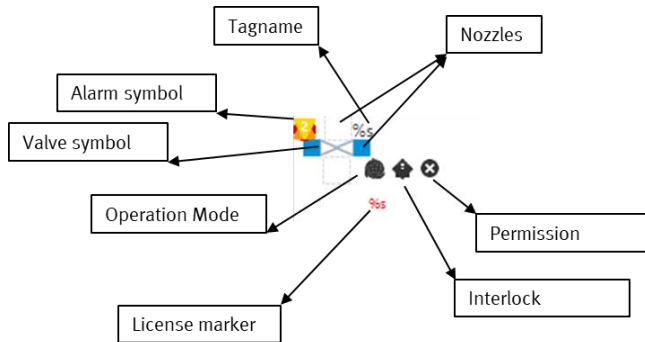


Fig. 61 Symbol hMonBinVlvValue

11.2.5.3 vMonBinVlvValue

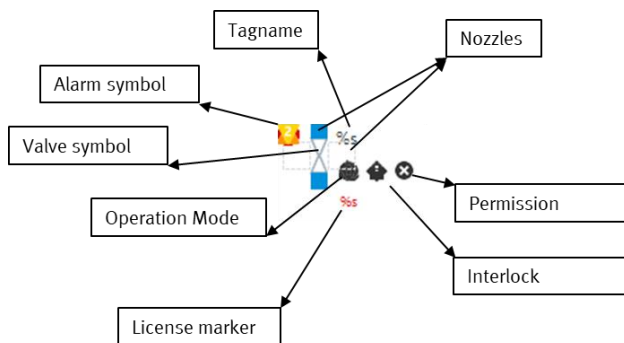


Fig. 62 Symbol vMonBinVlvValue

11.2.5.4 dldrMonThreeWayVlvValue, drdlMonThreeWayVlvValue, ldluMonThreeWayVlvValue, luldMonThreeWayVlvValue, rdruMonThreeWayVlvValue, rurdMonThreeWayVlvValue, ulurMonThreeWayVlvValue, urulMonThreeWayVlvValue

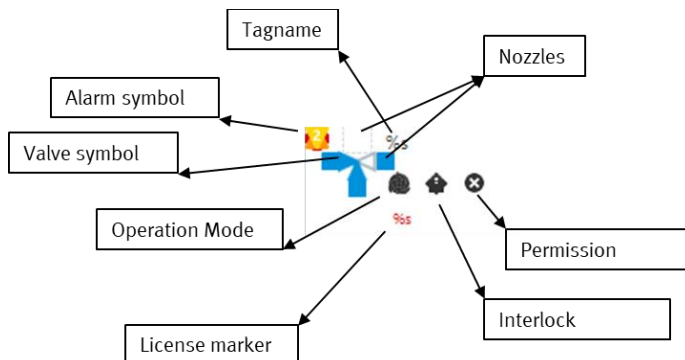


Fig. 63 Symbol MonThree-Way-Valves

11.2.6 Faceplates

11.2.6.1 Main

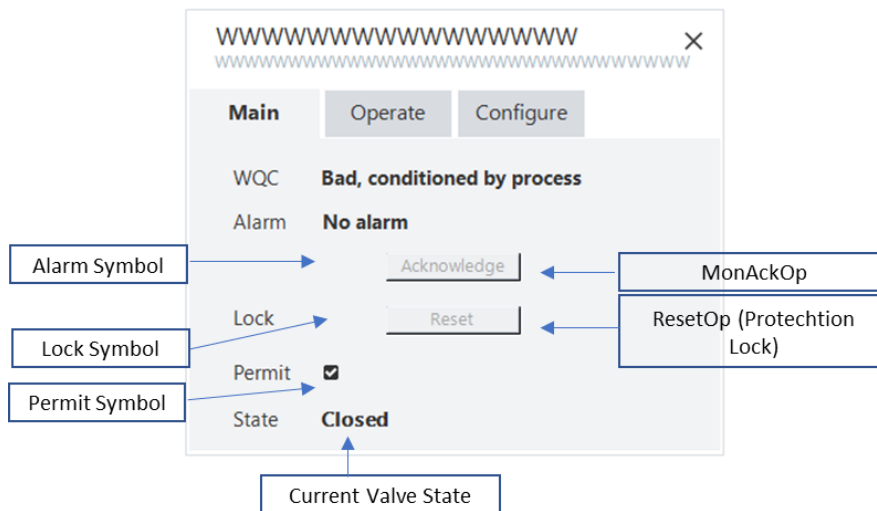


Fig. 64 MonBinVlv main faceplate

11.2.6.2 Operate

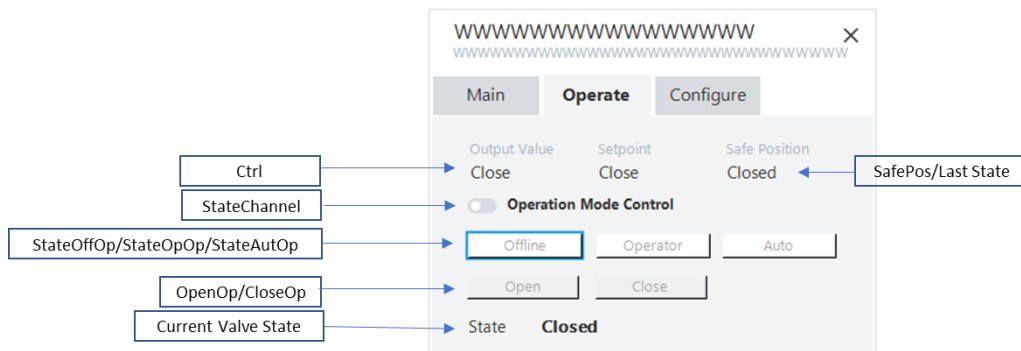


Fig. 65 MonBinVlv operate faceplate

11.2.6.3 Configure

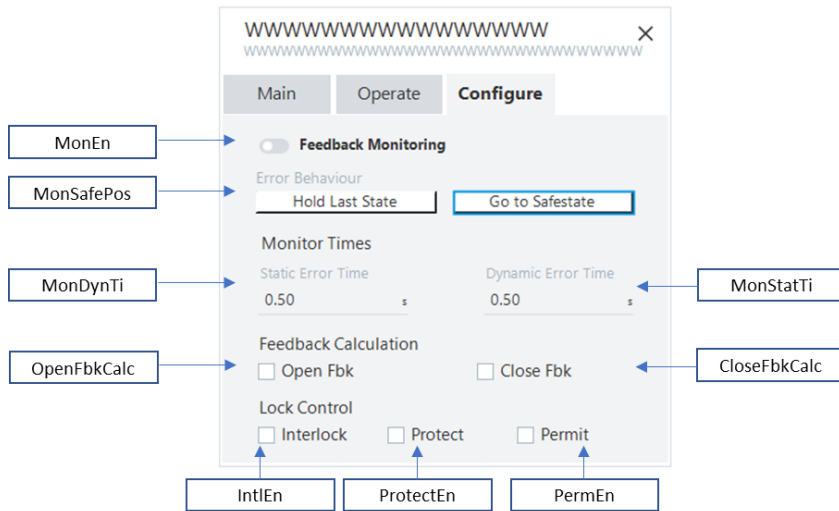


Fig. 66 MonBinVlv configure faceplate

12 Controllable valves

12.1 AnaVlv function block

12.1.1 Brief description

The AnaVlv function block (FB) provides the option of positioning a valve between a minimum and a maximum value. A setpoint position can be specified via an internal or a manual source. Feedback signals are also available which indicate the valve's actual state.

12.1.2 Basic functions

12.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

12.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 61 Worst quality code enumeration

12.1.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

12.1.3 Functions

12.1.3.1 “PosMan” and “PosInt” setpoint specification

The setpoint for the position can be specified via two sources. The setpoint can be specified directly at the FB via the “PosInt” input. If the setpoint will be selected via OPC UA or the faceplate, the entry field „Manual Setpoint“ is used. The source mode controls which of the two inputs will be used for positioning at any time. Upon initialisation, the internal source is active, i.e. “PosInt”. The mode of operation and handling of the source mode is explained in a separate section.

12.1.3.2 Setpoint limiting

Limit values can be entered for the position via the “PosMin” and “PosMax” inputs at the FB. If the specified setpoint (“PosInt” or “PosMan”) is greater than “PosMax” or less than “PosMin”, the setpoint is limited to the corresponding limit value. It must also be assured that “PosMin” is not greater than “PosMax”. Otherwise, there’s no guarantee that the conversion and the graphics in the visualisation will include sensible values.

12.1.3.3 Safety position “SafePos” and “SafePosEn”

The “SafePos” input can be used to specify whether the idle or safety position is minimum position “PosMin” (SafePos = FALSE) or maximum position “PosMax” (SafePos = TRUE). The “SafePosEn” can be used to specify whether or not the valve has a safety state. If there isn’t any safety state, the most recent state is always retained in the event of an error.

12.1.3.4 Valve control

The controllable valve can be positioned anywhere between “PosMin” and “PosMax”. Inputs are available at the FB (“OpenAut” and “CloseAut”) and variables are available via OPC UA / the faceplate (“OpenOp” and “CloseOp”) for controlling the valve. The corresponding control command is processed by the FB with the “PosInt” or “PosMan” setpoint value and read out to the “Pos” output.

Depending on the command, control commands are issued by setting the above-mentioned variables to TRUE. Input variables “OpenAut” and “CloseAut” are edge-triggered and react to rising edges. “OpenOp” and “CloseOp” are set back to FALSE after registration by the FB.

The following table shows how the output performs depending on the safety position and the control commands.

Control command	Outputs
Open	Pos = PosInt/PosMan
Close	Pos = PosMin
Open and close at the same time	Pos = SafePos

Tab. 62 Valve control commands

The FB can be operated in three states, namely offline, operator and automatic. The states are controlled via the operation mode. This mode is described in a separate section.

The following control commands and outputs are processed and set in the respective state:

- Offline mode: Pos = SafePos
- Operator mode: “OpenOp” and “CloseOp” control “Pos”, “SpMan”/“Splnt” are valid (can be changed via faceplate)
- Automatic mode: “OpenAut” and “CloseAut” control “Pos”, “SpMan”/“Splnt” are valid. With OpenAut, the setpoint value is continuously tracked.

12.1.3.5 Units of measure for position

A unit of measure can be entered for the position via the “PosUnit” input at the FB. The “UnitSettings” enumeration is used in order to specify a unit of measure. The unit of measure is only used for displays and has no influence on the conversion.

12.1.3.6 Scaling values for the visualisation

One minimum and one maximum variable (“PosScIMin” and “PosScIMax”) are used to set the display limits of each visual representation of the position. These variables are used exclusively for the representation of graphic elements in the visualisation. If the display limits are set incorrectly, they’re set automatically to the the position’s minimum and maximum values. Refer to the section entitled “FB-specific worst quality code” in this regard.

12.1.3.7 Readback

The “PosRbk” output reads the active input value (“PosMan” or “PosInt”) without taking limit values “PosMin” and “PosMax” into consideration.

12.1.3.8 Feedback signals (simulated, actual valve state)

Feedback signals are simulated automatically by the FB itself, when the “OpenFbkCalc” and/or “CloseFbkCalc” inputs are set to TRUE.

In the event that no simulation is desired, the “OpenFbkSignal” and/or “CloseFbkSignal” inputs can be linked to the valve’s actual feedback signals. Furthermore, “OpenFbkCalc” and “CloseFbkCalc” must be set to FALSE. When an end position is reached, the corresponding output at the FB (“OpenFbk” or “CloseFbk”) is set to TRUE. As a default setting, the correct feedback signals are processed via “OpenFbkSignal” and/or “CloseFbkSignal”.

Position feedback can be simulated as well. The “PosFbkCalc” input at the FB must be set to TRUE to this end. The calculated position is read out from the “PosFbk” output. If the “PosFbkCalc” is set to FALSE, feedback from the valve can be directly transmitted via the “PosFbkSignal” input at the FB and read out to the “PosFbk” output.

12.1.3.9 Lock functions

12.1.3.9.1 Permit

The permit function can be activated via the “PermEn” input (= TRUE). If the “Permit” input is set to TRUE, controller commands can be processed. If the “Permit” input is set to FALSE, control commands are no longer processed. Control commands (e.g. for valves) are no longer executed. No control outputs are set either (e.g. at the operators). The FB retains the most recent state – only the safety position/value can be approached. The FB then remains in this position until the “Permit” input is set back to TRUE.

12.1.3.9.2 Interlock

The interlock function can be activated via the “IntlEn” input (TRUE). If the “Interlock” input is set to TRUE, control commands can be processed. If the “Interlock” input is set to FALSE, no control commands are processed. The safety position/value is approached, if the FB has one. If not, the most recent state is retained. No control commands are processed until the “Interlock” input is set back to TRUE or the function is deactivated via “IntlEn”. When the “Interlock” input is once again set to TRUE, the FB remains in the safety state until a new control command is transmitted.

If the FB is in the safety state, “SafePosAct” is set to TRUE.

12.1.3.9.3 Protection lock

The protection lock function can be activated via the “ProtEn” input (TRUE). If the “Protect” input is set to TRUE, control commands can be processed. If the “Protect” input is set to FALSE, no control commands are processed. The safety position/value is approached, if the FB has one. If not, the most recent state is retained. Like the interlock function, no control commands are processed until the “Protect” input is set back to TRUE or the function is deactivated via “ProtEn”. The protection lock must be additionally reset. This is possible in the visualisation via the faceplate (“ResetOp”) or via the “ResetAut” input at the FB.

Note:

- The lock function is active in all states (offline, operator, automatic).
- “ResetAut” is edge-triggered and reacts to rising edges.

12.1.4 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
40d	16#28	2	Bad, conditioned by process	Offline mode it active
				Pos < PosMin ⇒ Pos = PosMin
				Pos > PosMax ⇒ Pos = PosMax
				PosMin > PosMax
				PosSclMax < PosSclMin ⇒ PosSclMax = PosMax ⇒ PosSclMin = PosSclMax
				PosSclMin < PosMin or PosSclMin > PosMax ⇒ PosSclMin = PosMin
				PosSclMax < PosMin or PosSclMax > PosMax ⇒ PosSclMax = PosMax
				PosSclMax – PosSclMin = 0 ⇒ PosSclMax = PosMax ⇒ PosSclMin = PosMin
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 63 AnaVlv specific worst quality code

12.1.5 Source mode

12.1.5.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (manual). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (*Int) the “SrcIntAct” output is set to TRUE. When the manual source is active (*Man), the “SrcManAct” output bit is set to TRUE.

The internal source is active as a default setting.

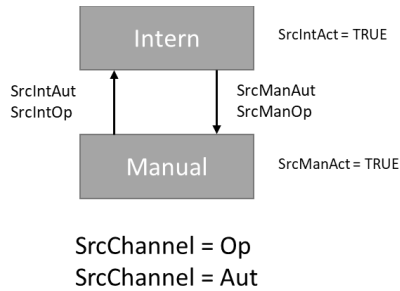


Fig. 67 Statemachine source mode

12.1.5.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutoOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcManAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAct” and “SrcManAct” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcManAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcManAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

12.1.5.3 Module-external access (OPC UA)

The “SrcIntOp” and “SrcManOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutoOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “SrcIntOp” (internal source) or “SrcManOp” (manual source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcManOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcManAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op

variables, requests are not processed by the FB although they're acknowledged. Invalid requests are not saved for a later point in time.

If both inputs ("SrcIntOp" and "SrcManOp") are set simultaneously, "SrcIntOp" has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable "SrcIntOp", "SrcManOp", "SrcIntAut" or "SrcManAut", the variable is set back to FALSE in order to acknowledge the request. For this reason, it's important to ensure that these inputs are not permanently assigned to the FB.

12.1.6 Operation mode

12.1.6.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode. Changes of state are only possible between offline and operator or operator and automatic. Changing directly from offline to automatic is not allowed.

The "StateChannel" input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the "StateOffAct" output is set to TRUE. When the operator mode is active "StateOpAct" is set to TRUE, and when the automatic mode is active "StateAutAct" is set to TRUE.

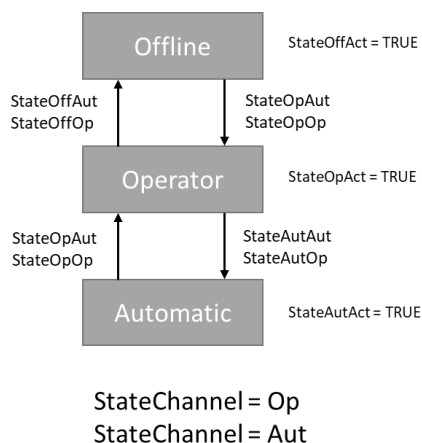


Fig. 68 Statemachine operation mode

12.1.6.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- “StateChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “StateOffAut” input (switch to offline mode), the “StateOpAut” input (switch to operator mode) or the “StateAutAut” input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

12.1.6.3 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.
- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the

variable is set back to FALSE in order to acknowledge the request. For this reason, it's important to ensure that these inputs are not permanently assigned to the FB.

12.1.7 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
SafePosEn	BOOL	TRUE	Enables Safe Position (1: Device has a safe position 0: Device has no safe position)
SafePos	BOOL	FALSE	Safe position (1: PosMax 0: PosMin)
OSLevel	BYTE	0	OS Level Variable

Tab. 64 AnaVlv persistent variables

12.1.8 Symbols

12.1.8.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Valve symbol	Grey: “CloseFbk” is active Blue: “OpenFbk” is active	
Operation Mode	 : offline mode is active  : operator mode is active  : automatic mode is active	See description of operation mode.
Interlock	: no interlock or protection lock is active  : interlock is active  : protection lock is active	See lock function description.
Permission	 : access permitted  : access denied	See lock function description.
Position setpoint Bargraph	Depending on the source mode, “PosInt” or “PosMan” is displayed as a blue bar.	“PosSclMin” and “PosSclMax” scale the bar graph.
Process value Bargraph	“PosFbk” is displayed as a blue bar.	“PosSclMin” and “PosSclMax” scale the bar graph.
Position setpoint	Depending on the source mode, “PosInt” or “PosMan” is displayed.	Max. 5 places before and 2 places after the decimal point are displayed.
Process value	“PosFbk” is displayed.	Max. 5 places before and 2 places after the decimal point are displayed.
Unit	U/M “PosUnit” is displayed for the setpoint value and the process value.	Only values from the “UnitSettings” enumeration are possible.
Nozzles	Nozzles for connecting logical lines (dashed nozzles) and pipes (blue nozzles).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 65 AnaVlv symbol description

12.1.8.2 hAnaVlvValue

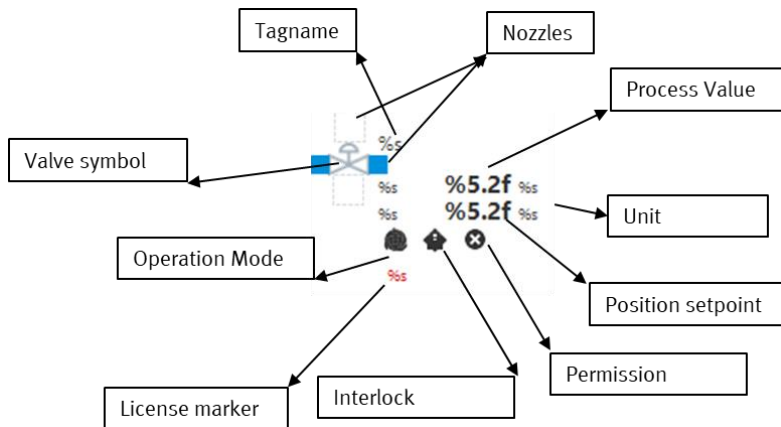


Fig. 69 Symbol hAnaVlvValue

12.1.8.3 vAnaVlvValue

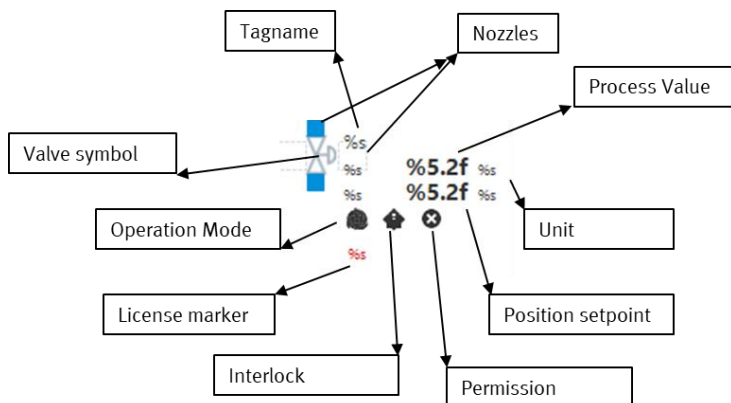


Fig. 70 Symbol vAnaVlvValue

12.1.8.4 hAnaVlvBargraph

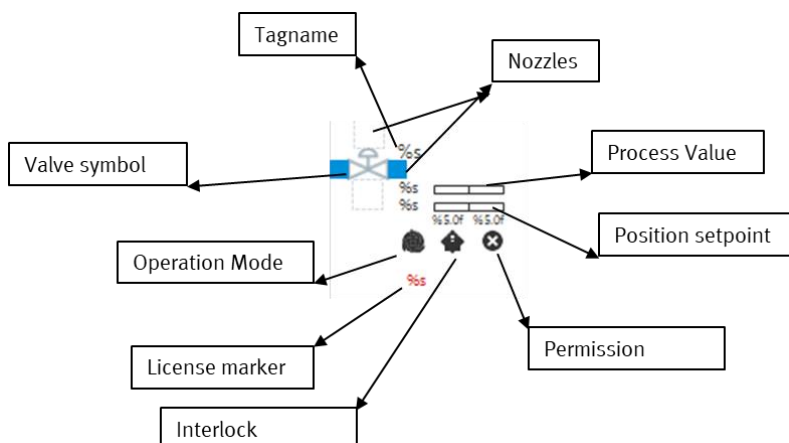


Fig. 71 Symbol hAnaVlvBargraph

12.1.8.5 vAnVlvBargraph

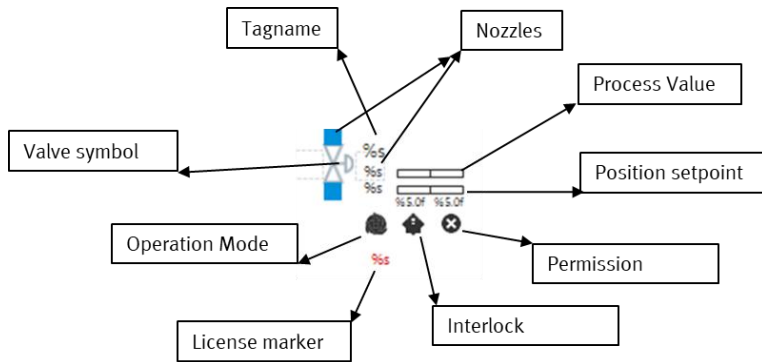


Fig. 72 Symbol vAnVlvBargraph

12.2 MonAnaVlv function block

12.2.1 Brief description

The MonAnaVlv function block (FB) expands the AnaVlv FB to include feedback monitoring for dynamic and static errors, as well as position monitoring.

12.2.2 Extended functions

12.2.2.1 Activate feedback monitoring

Feedback monitoring (static, dynamic and position monitoring) can be activated via the “MonEn” input at the FB (= TRUE). If “MonEn” is set to FALSE or if it’s set to FALSE during operation, all error outputs are reset. The alarm symbol also disappears from the visualisation and the error doesn’t need to be acknowledged.

12.2.2.2 “MonSafePos” input

The “MonSafePos” input at the FB can be used to specify whether the valve advances to the safety position in the event of an error (MonSafePos = TRUE) or the current state is retained (MonSafePos = FALSE). In either case, the SafePosAct output bit is set to TRUE in order to detect the exception state. As a default setting, travel to the safety position takes place in the event of an error.

If the most recent state should be retained after an error, or if the valve is already in the safety state, errors are not eliminated until corresponding feedback signals provide the correct values.

If the valve is set to the safety state after an error, the error is not eliminated until the valve returns to the state in which the error occurred and the feedback signals provide the correct values.

Example:

The valve is in the closed state and needs to be opened. After opening, “OpenFbk” is still set to FALSE. The valve is thus set to its safety state again (in this case closed). In order to eliminate the error, the valve must be opened again and “OpenFbk” must be set to TRUE. Only then are the error outputs (“StatErrAct” and “DynErrAct”) set back to FALSE.

12.2.2.3 Static errors

A static error has occurred when the feedback signal changes although no change has taken place at the controller.

The following static errors are possible:

- “OpenFbk” and “CloseFbk” are both TRUE
- State is open, “OpenFbk” is set to FALSE
- State is closed, “CloseFbk” is set to FALSE

In order to assure that a malfunction is not assessed as a lost or incorrect signal, the error state must persist for a certain duration in order for it to be evaluated as an error. This duration can be set via the “MonStatTi” input at the FB. The error is not recognised and the “StatErrAct” output is not set to TRUE unless the error state is still pending after this time has elapsed.

If a visualisation is used in CODESYS, the corresponding alarm appears in the symbol.

Note:

If travel to the safe position should take place in the event of an error, OpenAut and safe position would continuously cancel each other out in the automatic mode.

In this case, travel takes place to the safe position and the OpMode is set to operator, because intervention is necessary due to the error.

The FB can then be switched again via the faceplate or the inputs at the FB. If the error has not been eliminated, the operator mode is reactivated.

12.2.2.4 Dynamic errors

A dynamic error has occurred when the feedback signal doesn't change although a change has taken place at the controller.

The following dynamic errors are possible:

- State should change from open to closed – “CloseFbk” remains FALSE
- State should change from closed to open – “OpenFbk” remains FALSE

In order to assure that the feedback signal is not expected immediately (because a valve requires a certain amount of time to close and open), a duration can be set via the “MonDynTi” input at the FB. If the correct feedback signal is not detected within this period of time, the “DynErrAct” output is set to TRUE.

If a visualisation is used in CODESYS, the corresponding alarm appears in the symbol.

Note:

If travel to the safe position should take place in the event of an error, OpenAut and safe position would continuously cancel each other out in the automatic mode.

In this case, travel takes place to the safe position and the OpMode is set to operator, because intervention is necessary due to the error.

The FB can then be switched again via the faceplate or the inputs at the FB. If the error has not been eliminated, the operator mode is reactivated.

12.2.2.5 Position monitoring

If travel to a position takes place via the control commands open and close, the valve must reach the respective position within a certain amount of time. This duration can be set via the “MonPosTi” input at the FB. A tolerance value can also be entered via the “PosTolerance” input at the FB. This value is added to and subtracted from the setpoint position so that a tolerance range can be calculated, within which the position is deemed reached. If the valve does not advance to within the tolerance range within the specified period of time, the “MonPosErr” is set to TRUE.

If a visualisation is used in CODESYS, the corresponding alarm appears in the symbol.

Note:

In the event of a position error, travel to the safe position does not take place and only the corresponding error variables are set.

12.2.2.6 Alarm priority

The following priorities are adhered to for the alarm display in the visualisation:

- Failed to open (top priority)
- Failed to close
- Both feedbacks active
- Open feedback lost
- Closed feedback lost
- Failed to reach position
- No alarm

12.2.2.7 Durations “MonStatTi”, “MonDynTi” and “MonPosTi”

The “MonStatTi”, “MonDynTi” and “MonPosTi” variables are specified in seconds and are set to 0.5 s as a default value. This duration can be changed via the input at the FB or at the faceplate. Values of less than 0 are converted to positive numbers. For example, if -5 s is entered, time is set to 5 s.

12.2.2.8 Acknowledging alarms

If no errors are pending anymore, the corresponding output bits are set back to FALSE. However, the error in the symbol must be reset by operator via the faceplate (“MonAckOp”) or via the “MonAckAut” input at the FB. Visibility of the alarm symbol is described in the table.

Limit value violation still pending	Error acknowledged	Symbol
Yes	No	Blinking
Yes	Yes	Visible
No	No	Blinking
No	Yes	Invisible – no error

Tab. 66 Alarm symbol states

Note:

- The “MonAckAut” input is edge-triggered and only reacts to rising edges.
- When acknowledged, all currently pending alarms are acknowledged simultaneously.
- Multiple limit errors may also be pending – only the highest priority is displayed in the visualisation.

12.2.3 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
40d	16#28	2	Bad, conditioned by process	Offline mode it active
				Pos < PosMin ⇒ Pos = PosMin
				Pos > PosMax ⇒ Pos = PosMax
				PosMin > PosMax
				PosSclMax < PosSclMin ⇒ PosSclMax = PosMax ⇒ PosSclMin = PosSclMax
				PosSclMin < PosMin or PosSclMin > PosMax ⇒ PosSclMin = PosMin
				PosSclMax < PosMin or PosSclMax > PosMax ⇒ PosSclMax = PosMax
				PosSclMax – PosSclMin = 0 ⇒ PosSclMax = PosMax ⇒ PosSclMin = PosMin
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 67 MonAnaVlv spezific worst quality code

12.2.4 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application** -> **Add Object** -> **Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
SafePosEn	BOOL	TRUE	Enables Safe Position (1: Device has a safe position 0: Device has no safe position)
SafePos	BOOL	FALSE	Safe position (1: PosMax 0: PosMin)
OSLevel	BYTE	0	OS Level Variable
MonEn	BOOL	FALSE	Enables dynamic, static and position Feedback Monitoring (1: enabled 0: disabled)
MonSafePos	BOOL	TRUE	Enables Monitor Safe Position (1: go to Safe Position 0: Hold last state)
MonStatTi	REAL	0.5	Monitor Time for uncontrolled changes [s], only positive values
MonDynTi	REAL	0.5	Monitor Time for controlled changes [s], only positive values
PosTolerance	REAL	0	Position Tolerance Parameter for calculation [PosUnit]
MonPosTi	REAL	0.5	Monitor Time until position is reached [s], only positive values

Tab. 68 MonAnaVlv persistent variables

12.2.5 Symbols

12.2.5.1 Description

Element	Description	Rule
Tagname	Name of the unit of measure – indicated in the visualisation.	Max. 16 characters.
Valve symbol	Grey: “CloseFbk” is active Blue: “OpenFbk” is active	
Alarm symbol	 : alarm is active (top priority)	See description of function block and alarm function.
Operation Mode	 : offline mode is active  : operator mode is active  : automatic mode is active	See description of operation mode.
Interlock	 : no interlock or protection lock is active  : interlock is active  : protection lock is active	See lock function description.
Permission	 : access permitted  : access denied	See lock function description.
Position setpoint Bargraph	Depending on the source mode, “PosInt” or “PosMan” is displayed as a blue bar.	“PosSclMin” and “PosSclMax” scale the bar graph.
Process value Bargraph	“PosFbk” is displayed as a blue bar in the bar graph.	“PosSclMin” and “PosSclMax” scale the bar graph.
Position setpoint	Depending on the source mode, “PosInt” or “PosMan” is displayed.	Max. 5 places before and 2 places after the decimal point are displayed.
Process value	“PosFbk” is displayed.	Max. 5 places before and 2 places after the decimal point are displayed.
Unit	U/M “PosUnit” is displayed for the setpoint value and the process value.	Only values from the “UnitSettings” enumeration are possible.
Unit	U/M “PosUnit” is displayed for the setpoint value and the process value.	Only values from the “UnitSettings” enumeration are possible.
Nozzles	Nozzles for connecting logical lines (dashed nozzles) and pipes (blue nozzles).	
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 69 MonAnaVlv symbol description

12.2.5.2 hMonAnaVlvValue

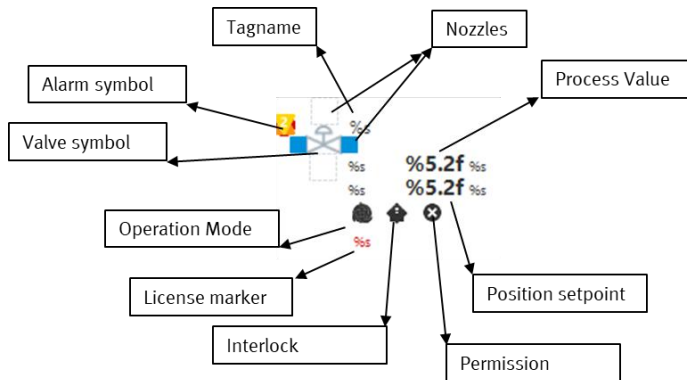


Fig. 73 Symbol hMonAnaVlvValue

12.2.5.3 vMonAnaVlvValue

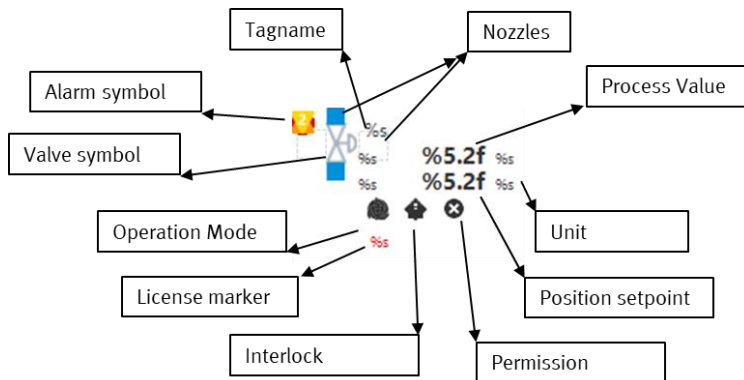


Fig. 74 Symbol vMonAnaVlvValue

12.2.5.4 hMonAnaVlvBargraph

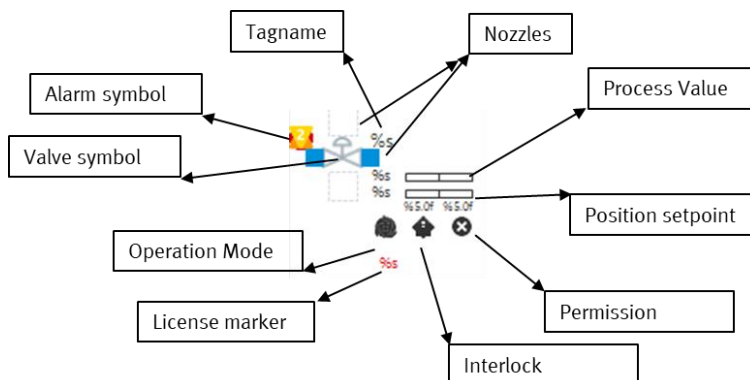


Fig. 75 Symbol hMonAnaVlvBargraph

12.2.5.5 vMonAnaVlvBargraph

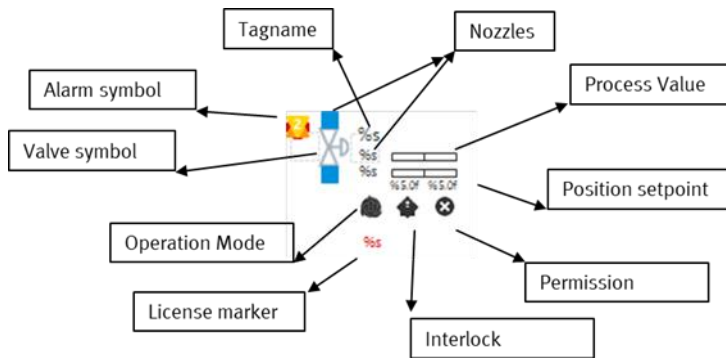


Fig. 76 Symbol vMonAnaVlvBargraph

12.2.6 Faceplates

12.2.6.1 Main

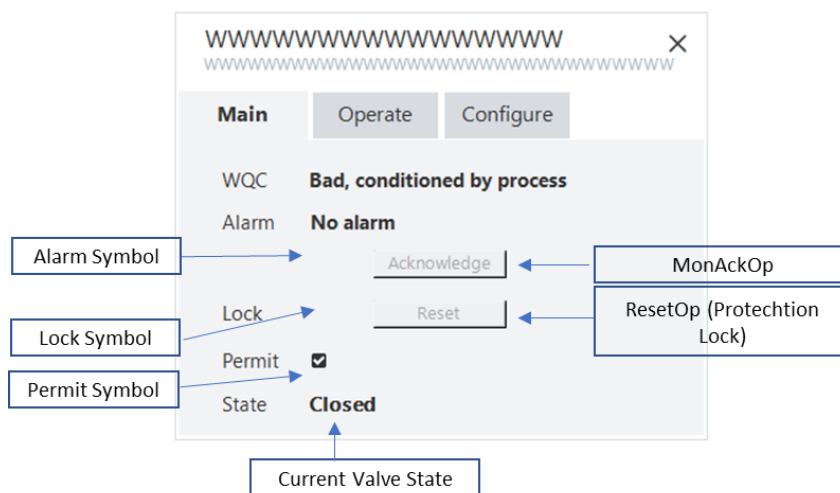


Fig. 77 MonAnaVlv main faceplate

12.2.6.2 Operate

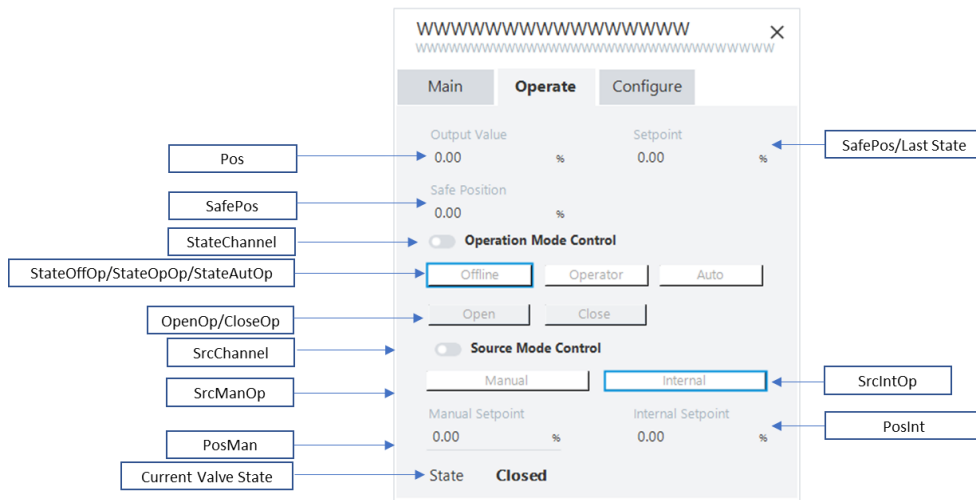


Fig. 78 MonAnaVlv operate faceplate

12.2.6.3 Configure

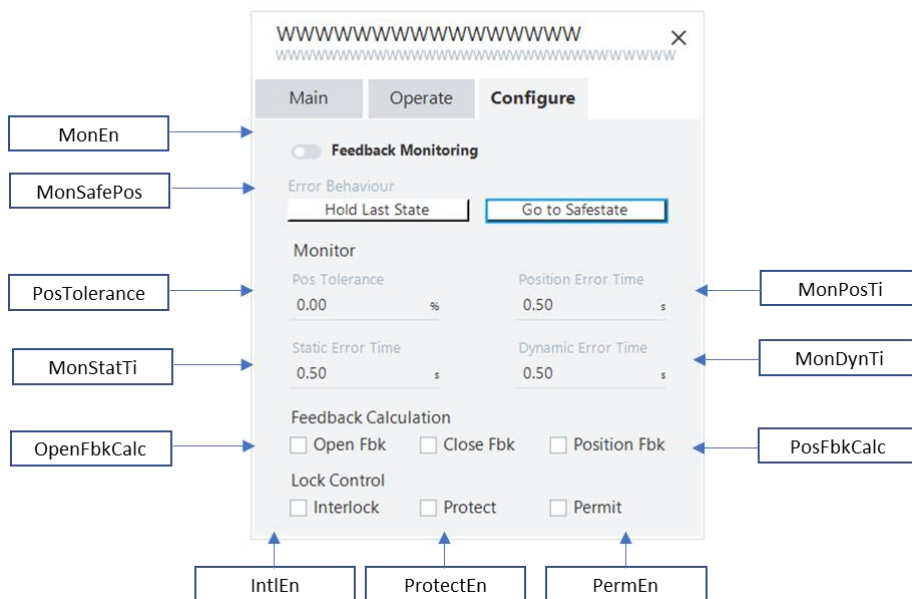


Fig. 79 MonAnaVlv configure faceplate

13 Bistable motors

13.1 BinDrv function block

13.1.1 Brief description

The BinDrv function block (FB) provides the option of switching a motor on or off, as well as specifying its direction of rotation. Control commands can be set either directly at the FB (Aut variables) or via a Faceplate / OPC UA (Op variables). Feedback signals are also available which indicate the drive's actual state.

13.1.2 Basic functions

13.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

13.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 70 Worst quality code enumeration

13.1.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

13.1.3 Functions

13.1.3.1 Motor controller

The motor can be run forward or in reverse, or can be stopped. In order to control the motor, inputs are available at the FB (“FwdAut”, “RevAut” and “StopAut”) and variables are available via OPC UA / faceplate (“FwdOp”, “RevOp” und “StopOp”). Forward and reverse operation must be activated respectively via an input at the FB, namely “FwdEn” or “RevEn” (= TRUE). The control command is processed by the FB and read out to the corresponding output, namely “FwdCtrl” or “RevCtrl”.

Depending on the command, control commands are issued by setting the above-mentioned variables to TRUE. Input variables “FwdAut”, “RevAut” and “StopAut” are edge-triggered and react to rising edges. “FwdOp”, “RevOp” and “StopOp” are set back to FALSE after registration by the FB.

The following table shows how the outputs perform depending on the safety position and the control commands.

Control command	Outputs
Stop	FwdCtrl = 0 RevCtrl = 0
Fwd	FwdCtrl = 1 RevCtrl = 0
Rev	FwdCtrl = 0 RevCtrl = 1
Fwd, Rev, Stop at the same time	FwdCtrl = 0 RevCtrl = 0
Fwd, Rev at the same time	Hold last state

Tab. 71 Motor control commands

The FB can be operated in three states, namely offline, operator and automatic. The states are controlled via the operation mode. This mode is described in a separate section.

The following control commands and outputs are processed and set in the respective state:

- Offline mode: FwdCtrl = 0, RevCtrl = 0
- Operator mode: “StopOp”, “FwdOp” and “RevOp” control “FwdCtrl” and “RevCtrl” (can be changed via faceplate)
- Automatic mode: “StopAut”, “FwdAut” and “RevAut” control “FwdCtrl” and “RevCtrl” (inputs at the FB)

13.1.3.2 Safety position “SafePos”

The “SafePos” input can be used to specify whether the safety state is running (SafePos = TRUE) or stopped (SafePos = FALSE).

The following points must be observed in order to assure that no errors occur in the safety state:

- If the safety state is running, either forward or reverse (“FwdEn” or “RevEn”) must be activated.
- If the safety state is running and the current state is also running, this state is retained in the event of an error
- If the safety state is running and the current state is stop, forward running is executed if it’s activated. If it’s not activated, checking takes place to determine whether or not reverse running is activated. If this isn’t activated either, the stop state is retained.

The various states are summarised in the following table:

Safety position	Current state	Error state
SafePos = TRUE (running)	FwdCtrl = 1 RevCtrl = 0	FwdCtrl = 1 RevCtrl = 0
SafePos = TRUE (running)	FwdCtrl = 0 RevCtrl = 1	FwdCtrl = 0 RevCtrl = 1
SafePos = TRUE (running)	FwdCtrl = 0 RevCtrl = 0	FwdEn = 1 -> FwdCtrl = 1 RevEn = 1 -> RevCtrl = 1 FwdEn = 0 and RevEn = 0 -> FwdCtrl = 0, RevCtrl = 0
SafePos = FALSE (stopped)	FwdCtrl = 1 RevCtrl = 0	FwdCtrl = 0 RevCtrl = 0
SafePos = FALSE (stopped)	FwdCtrl = 0 RevCtrl = 1	FwdCtrl = 0 RevCtrl = 0
SafePos = FALSE (stopped)	FwdCtrl = 0 RevCtrl = 0	FwdCtrl = 0 RevCtrl = 0

Tab. 72 Motor safety position states

13.1.3.3 Feedback signals (simulated, actual motor state)

Feedback signals are simulated automatically by the FB itself, when the “FwdFbkCalc” and/or “RevFbkCalc” inputs are set to TRUE.

In the event that no simulation is desired, the “FwdFbkSignal” and/or “RevFbkSignal” inputs can be linked to the motor’s actual feedback signals. Furthermore, “FwdFbkCalc” and “RevFbkCalc” must be set to FALSE. When an end position is reached, the corresponding output at the FB (“FwdFbk” or “RevFbk”) is set to TRUE. If both feedbacks are set to FALSE, the stop state has been reached. As a default setting, the correct feedback signals are processed via “FwdFbkSignal” and/or “RevFbkSignal”.

13.1.3.4 Trip

The trip signal can be forwarded to the FB via the “TripFbk” input at the FB. This signal is forwarded to the “Trip” output for further processing.

13.1.3.5 Lock functions

13.1.3.5.1 Permit

The permit function can be activated via the “PermEn” input (= TRUE). If the “Permit” input is set to TRUE, controller commands can be processed. IF the “Permit” input is set to FALSE, control commands are no longer processed. Control commands (e.g. for valves) are no longer executed. No control outputs are set either (e.g. at the operators). The FB retains the most recent state – only the safety position/value can be approached. The FB then remains in this position until the “Permit” input is set back to TRUE.

Note:

If the safe position is set to “running”, switching between forward and reverse running is possible via “Permit”. Only stopping the motor is not possible. If this type of performance is not desired, one of the directions can be deactivated via “FwdEn” and “RevEn” respectively.

13.1.3.5.2 Interlock

The interlock function can be activated via the “IntlEn” input (TRUE). If the “Interlock” input is set to TRUE, control commands can be processed. IF the “Interlock” input is set to FALSE, no control commands are processed. The safety position/value is approached, if the FB has one. If not, the most recent state is retained. No control commands are processed until the “Interlock” input is set back to TRUE or the function is deactivated via “IntlEn”. When the “Interlock” input is once again set to TRUE, the FB remains in the safety state until a new control command is transmitted.

If the FB is in the safety state, “SafePosAct” is set to TRUE.

13.1.3.5.3 Protection lock

The protection lock function can be activated via the “ProtEn” input (TRUE). If the “Protect” input is set to TRUE, control commands can be processed. IF the “Protect” input is set to FALSE, no control commands are processed. The safety position/value is approached, if the FB has one. If not, the most recent state is retained. Like the interlock function, no control commands are processed until the “Protect” input is set back to TRUE or the function is deactivated via “ProtEn”. The protection lock must be additionally reset. This is possible in the visualisation via the faceplate (“ResetOp”) or via the “ResetAut” input at the FB.

Note:

- The lock function is active in all states (offline, operator, automatic).
- “ResetAut” is edge-triggered and reacts to rising edges.

13.1.4 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
40d	16#28	2	Bad, conditioned by process	Offline mode it active
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 73 BinDrv specific worst quality code

13.1.5 Operation mode

13.1.5.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode. Changes of state are only possible between offline and operator or operator and automatic. Changing directly from offline to automatic is not allowed.

The “StateChannel” input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the “StateOffAct” output is set to TRUE. When the operator mode is active “StateOpAct” is set to TRUE, and when the automatic mode is active “StateAutAct” is set to TRUE.

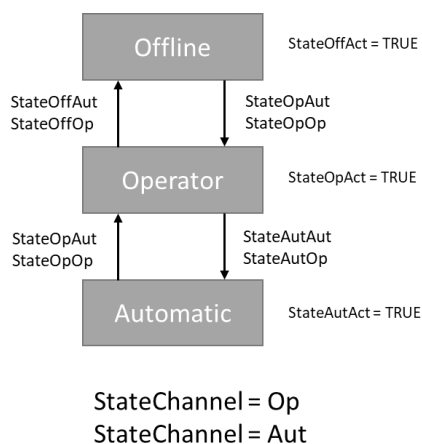


Fig. 80 Statemachine operation mode

Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- “StateChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “StateOffAut” input (switch to offline mode), the “StateOpAut” input (switch to operator mode) or the “StateAutAut” input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

13.1.5.2 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.
- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

13.1.6 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
SafePos	BOOL	FALSE	Safe Position (1: running 0: stopped)
OSLevel	BYTE	0	OS Level Variable

Tab. 74 BinDrv persistent variables

13.1.7 Symbols

13.1.7.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Drive symbol	Grey: motor is stopped Blue: motor is running	
Operation Mode	 : offline mode is active  : operator mode is active  : automatic mode is active	See description of operation mode.
Interlock	 : no interlock or protection lock is active  : interlock is active  : protection lock is active	See lock function description.
Permission	 : access permitted  : access denied	See lock function description.
Drehrichtung	 : motor running forward  : motor running in reverse	No symbol is displayed when the motor is stopped.
Nozzles	Nozzles for connecting logical lines (dashed nozzles), measurement lines (nozzle with solid line) and pipes (blue nozzles).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 75 BinDrv symbol description

13.1.7.2 dBinDrvPumpValue/uBinDrvPumpValue/rBinDrvPumpValue/lBinDrvPumpValue

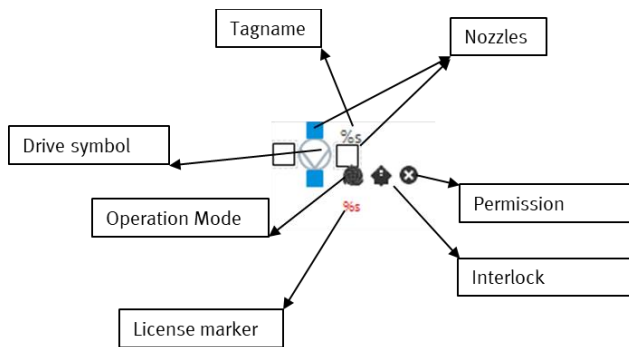


Fig. 81 Symbols BinDrv Pump

13.1.7.3 hBinDrvValue

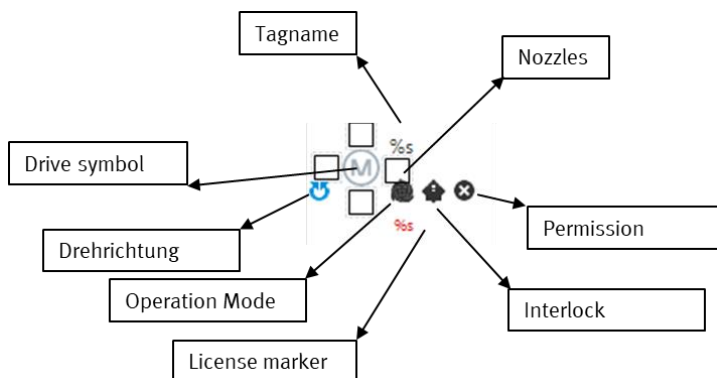


Fig. 82 Symbol hBinDrvValue

13.2 MonBinDrv function block

13.2.1 Brief description

The MonBinDrv function block (FB) expands the BinDrv FB to include feedback monitoring for dynamic and static errors.

13.2.2 Extended functions

13.2.2.1 Activate feedback monitoring

Feedback monitoring (static and dynamic) can be activated via the “MonEn” input at the FB (= TRUE). If “MonEn” is set to FALSE or if it’s set to FALSE during operation, all error outputs are reset. The alarm symbol also disappears from the visualisation and the error doesn’t need to be acknowledged.

13.2.2.2 “MonSafePos” input

The “MonSafePos” input at the FB can be used to specify whether the motor enters the safety state in the event of an error (MonSafePos = TRUE) or the current state is retained (MonSafePos = FALSE). In either case, the SafePosAct output bit is set to TRUE in order to detect the exception state. As a default setting, travel to the safety position takes place in the event of an error.

If the most recent state should be retained after an error, or if the motor is already in the safety state, errors are not eliminated until corresponding feedback signals provide the correct values.

If the motor is set to the safety state after an error, the error is not eliminated until after returning to the state in which the error occurred and the feedback signals provide the correct values.

Example:

The motor is stopped and needs to be started. After starting, “FwdFbk” is still set to FALSE. The motor is thus set to its safety state again (in this case stopped). In order to eliminate the error, the motor must be started again and “FwdFbk” must be set to TRUE. Only then are the error outputs (“StatErrAct” and “DynErrAct”) set back to FALSE.

13.2.2.3 Static errors

A static error has occurred when the feedback signal changes although no change has taken place at the controller.

The following static errors are possible:

- “FwdFbk” and “RevFbk” are both TRUE.
- State is running, “FwdFbk”/ “RevFbk” is switched to FALSE.
- State is stopped, “FwdFbk”/ “RevFbk” is switched to TRUE.

In order to assure that a malfunction is not assessed as a lost or incorrect signal, the error state must persist for a certain duration in order for it to be evaluated as an error. This duration can be set via the “MonStatTi” input at the FB. The error is not recognised and the “StatErrAct” output is not set to TRUE unless the error state is still pending after this time has elapsed.

If a visualisation is used in CODESYS, the corresponding alarm appears in the symbol.

13.2.2.4 Dynamic errors

A dynamic error has occurred when the feedback signal doesn't change although a change has taken place at the controller.

The following dynamic errors are possible:

- State should change from running to stopped – “FwdFbk” remains TRUE
- State should change from stopped to running – “FwdFbk” remains FALSE
- State should change from forward running to reverse running – “RevFbk” remains FALSE

In order to assure that the feedback signal is not expected immediately (because a motor requires a certain amount of time to start up or stop), a duration can be set via the “MonDynTi” input at the FB. If the correct feedback signal is not detected within this period of time, the “DynErrAct” output is set to TRUE.

If a visualisation is used in CODESYS, the corresponding alarm appears in the symbol.

13.2.2.5 Alarm priority

The following priorities are adhered to for the alarm display in the visualisation:

- Both feedbacks active (top priority)
- Forward feedback lost
- Reverse feedback lost
- Stop feedback lost
- Failed to run forward
- Failed to run in reverse
- Failed to stop
- No alarm

13.2.2.6 Durations “MonStatTi” and “MonDynTi”

The “MonStatTi” and “MonDynTi” variables are specified in seconds and are set to 0.5 s as a default value. This duration can be changed via the input at the FB or at the faceplate. Values of less than 0 are converted to positive numbers. For example, if -5 s is entered, time is set to 5 s.

13.2.2.7 Acknowledging alarms

If no errors are pending anymore, the corresponding output bits are set back to FALSE. However, the error in the symbol must be reset by operator via the faceplate (“MonAckOp”) or via the “MonAckAut” input at the FB. Visibility of the alarm symbol is described in the table.

Limit value violation still pending	Error acknowledged	Symbol
Yes	No	Blinking
Yes	Yes	Visible
No	No	Blinking
No	Yes	Invisible – no error

Tab. 76 Alarm symbol states

Note:

- The “MonAckAut” input is edge-triggered and only reacts to rising edges.
- When acknowledged, all currently pending alarms are acknowledged simultaneously.
- Multiple limit errors may also be pending – only the highest priority is displayed in the visualisation.

13.2.3 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
40d	16#28	2	Bad, conditioned by process	Offline mode it active
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 77 MonBinDrv specific worst quality code

13.2.4 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It’s even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
SafePos	BOOL	FALSE	Safe Position (1: running 0: stopped)
OSLevel	BYTE	0	OS Level Variable

Variable	Type	Default	Description
MonEn	BOOL	FALSE	Enables dynamic, static and position Feedback Monitoring (1: enabled 0: disabled)
MonSafePos	BOOL	TRUE	Enables Monitor Safe Position (1: go to Safe Position 0: Hold last state)
MonStatTi	REAL	0.5	Monitor Time for uncontrolled changes [s], only positive values
MonDynTi	REAL	0.5	Monitor Time for controlled changes [s], only positive values

Tab. 78 MonBinDrv persistent variables

13.2.5 Symbols

13.2.5.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters
Alarm symbols	 : alarm is active	See description of function block and alarm function.
Drive symbol	Grey: motor is stopped Blue: motor is running	
Operation Mode	 : offline mode is active  : operator mode is active  : automatic mode is active	See description of operation mode.
Interlock	 : no interlock or protection lock is active  : interlock is active  : protection lock is active	See lock function description.
Permission	 : access permitted  : access denied	See lock function description.
Drehrichtung	 : motor running forward  : motor running in reverse	No symbol is displayed when the motor is stopped.
Nozzles	Nozzles for connecting logical lines (dashed nozzles), measurement lines (nozzle with solid line) and pipes (blue nozzles).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 79 MonBinDrv dymbol description

13.2.5.2 dMonBinDrvPumpValue/uMonBinDrvPumpValue/rMonBinDrvPumpValue/lMonBinDrvPumpValue

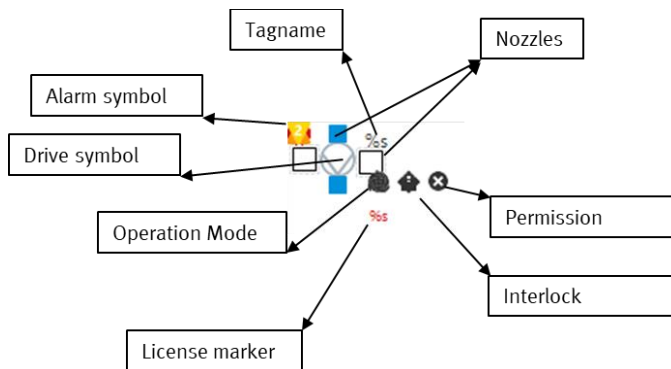


Fig. 83 Symbol MonBinDrv Pump

13.2.5.3 hMonBinDrvValue

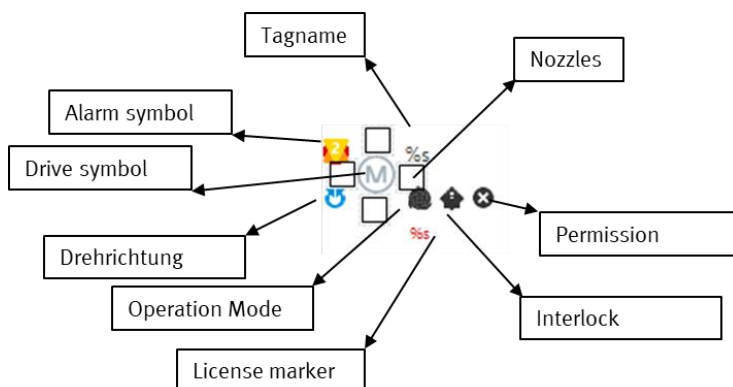


Fig. 84 Symbol hMonBinDrvValue

13.2.6 Faceplates

13.2.6.1 Main

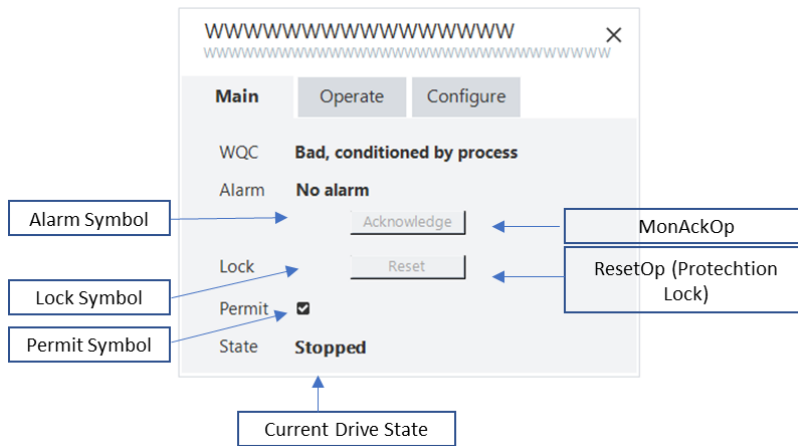


Fig. 85 MonBinDrv main faceplate

13.2.6.2 Operate

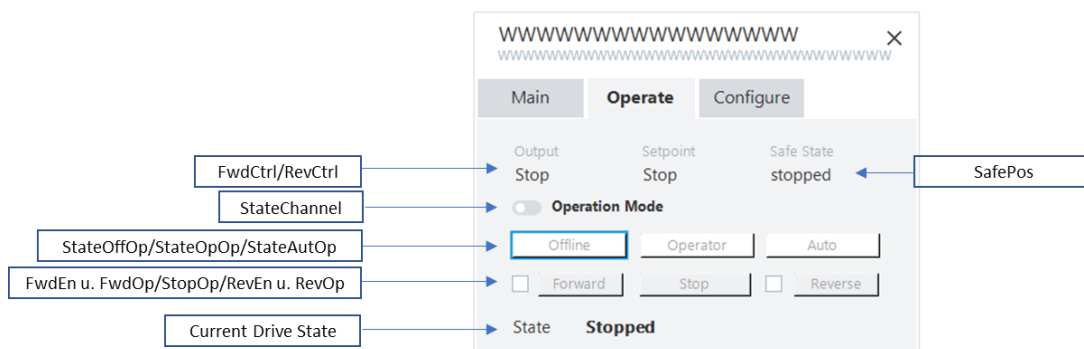


Fig. 86 MonBinDrv operate faceplate

13.2.6.3 Configure

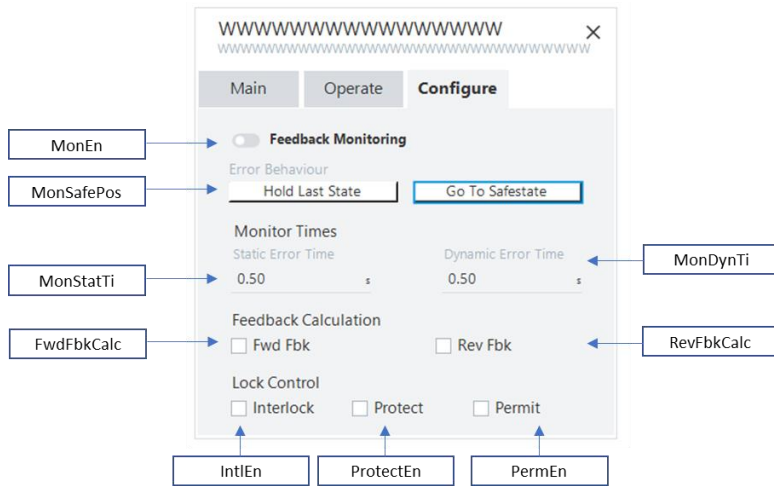


Fig. 87 MonBinDrv configure faceplate

14 Controllable motors

14.1 AnaDrv function block

14.1.1 Brief description

The AnaDrv function block (FB) provides the option of switching a motor on or off, as well as specifying its direction of rotation and a variable speed. Control commands can be set either directly at the FB (Aut variables) or via a faceplate / OPC UA (Op variables). Feedback signals are also available which indicate the drive's actual state.

14.1.2 Basic functions

14.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

14.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 80 Worst quality code enumeration

14.1.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

14.1.3 Functions

14.1.3.1 Speed setpoint specifications “RpmMan” and “RpmInt”

When the motor is started, a speed can be additionally selected. The setpoint for speed can be specified via two sources. The setpoint can be specified directly at the FB via the “RpmInt” input. If the setpoint will be selected via OPC UA or the faceplate, the entry field „Manual Setpoint“ is used. The source mode controls which of the two inputs will be used for speed at any time. Upon initialisation, the internal source is active, i.e. “RpmInt”. The mode of operation and handling of the source mode is explained in a separate section.

The selected setpoint value is read out to the “Rpm” output in consideration of the limit values.

14.1.3.2 Setpoint limiting, “RpmInt” and “RpmMan”

Limit values can be entered for speed via the “RpmMin” and “RpmMax” inputs at the FB. If the specified setpoint (“RpmInt” or “RpmMan”) is greater than “RpmMax” or less than “RpmMin”, the setpoint is limited to the corresponding limit value. It must also be assured that “RpmMin” is not greater than “RpmMax”. Otherwise, there’s no guarantee that the conversion and the graphics in the visualisation will include sensible values.

14.1.3.3 Unit of measure, “RpmInt”, “RpmMan” und “Rpm”

A unit of measure can be entered for speed via the “RpmUnit” input at the FB. The “UnitSettings” enumeration is used in order to specify a unit of measure. The unit of measure is only used for displays and has no influence on the conversion.

14.1.3.4 Motor controller

The motor can be run forward or in reverse, or can be stopped. In order to control the motor, inputs are available at the FB (“FwdAut”, “RevAut” and “StopAut”) and variables are available via OPC UA / faceplate (“FwdOp”, “RevOp” and “StopOp”). Forward and reverse operation must be activated respectively via an input at the FB, namely “FwdEn” or “RevEn” (= TRUE). The control command is processed by the FB and read out to the corresponding output, namely “FwdCtrl” or “RevCtrl”.

Depending on the command, control commands are issued by setting the above-mentioned variables to TRUE. Input variables “FwdAut”, “RevAut” and “StopAut” are edge-triggered and react to rising edges. “FwdOp”, “RevOp” and “StopOp” are set back to FALSE after registration by the FB.

The following table shows how the outputs perform depending on the safety position and the control commands.

Control command	Outputs
Stop	FwdCtrl = 0 RevCtrl = 0
Fwd	FwdCtrl = 1 RevCtrl = 0
Rev	FwdCtrl = 0 RevCtrl = 1
Fwd, Rev, Stop at the same time	FwdCtrl = 0 RevCtrl = 0
Fwd, Rev at the same time	Hold last state

Tab. 81 AnaDrv control commands

The FB can be operated in three states, namely offline, operator and automatic. The states are controlled via the operation mode. This mode is described in a separate section.

The following control commands and outputs are processed and set in the respective state:

- Offline mode: FwdCtrl = 0, RevCtrl = 0
- Operator mode: “StopOp”, “FwdOp” and “RevOp” control “FwdCtrl” and “RevCtrl” (can be changed via faceplate)
- Automatic mode: “StopAut”, “FwdAut” and “RevAut” control “FwdCtrl” and “RevCtrl” (inputs at the FB)

14.1.3.5 Safety position “SafePos”

The “SafePos” input can be used to specify whether the safety state is running (SafePos = TRUE) or stopped (SafePos = FALSE).

The following points must be observed in order to assure that no errors occur in the safety state:

- If the safety state is running, either forward or reverse (“FwdEn” or “RevEn”) must be activated.
- If the safety state is running and the current state is also running, this state is retained in the event of an error
- If the safety state is running and the current state is stop, forward running is executed if it’s activated. If it’s not activated, checking takes place to determine whether or not reverse running is activated. If this isn’t activated either, the stop state is retained.

The various states are summarised in the following table:

Safety position	Current state	Error state
SafePos = TRUE (running)	FwdCtrl = TRUE RevCtrl = FALSE	FwdCtrl = TRUE RevCtrl = FALSE
SafePos = TRUE (running)	FwdCtrl = FALSE RevCtrl = TRUE	FwdCtrl = FALSE RevCtrl = TRUE
SafePos = TRUE (running)	FwdCtrl = FALSE RevCtrl = FALSE	FwdEn = TRUE -> FwdCtrl = TRUE RevEn = TRUE -> RevCtrl = TRUE FwdEn = FALSE and RevEn = FALSE-> FwdCtrl = FALSE, RevCtrl = FALSE
SafePos = FALSE (stopped)	FwdCtrl = TRUE RevCtrl = FALSE	FwdCtrl = FALSE RevCtrl = FALSE
SafePos = FALSE (stopped)	FwdCtrl = FALSE RevCtrl = TRUE	FwdCtrl = FALSE RevCtrl = FALSE
SafePos = FALSE (stopped)	FwdCtrl = FALSE RevCtrl = FALSE	FwdCtrl = FALSE RevCtrl = FALSE

Tab. 82 AnaDrv safety position states

14.1.3.6 Feedback signals (simulated, actual motor state)

Feedback signals are simulated automatically by the FB itself, when the “FwdFbkCalc” and/or “RevFbkCalc” inputs are set to TRUE.

In the event that no simulation is desired, the “FwdFbkSignal” and/or “RevFbkSignal” inputs can be linked to the motor’s actual feedback signals. Furthermore, “FwdFbkCalc” and “RevFbkCalc” must be set to FALSE. When an end position is reached, the corresponding output at the FB (“FwdFbk” or “RevFbk”) is set to TRUE. If both feedbacks are set to FALSE, the stop state has been reached. As a default setting, the correct feedback signals are processed via “FwdFbkSignal” and/or “RevFbkSignal”.

The feedback signal can also be simulated for speed via an input at the “RpmFbkCalc” FB (=TRUE). If the feedback signal from the motor will be used, the actual signal can be linked via the input at the “RpmFbkSignal” FB. The “RpmFbkCalc” input must be set to FALSE to this end.

14.1.3.7 Trip

The trip signal can be forwarded to the FB via the “TripFbk” input at the FB. This signal is forwarded to the “Trip” output for further processing.

14.1.3.8 Lock functions

14.1.3.8.1 Permit

The permit function can be activated via the “PermEn” input (= TRUE). If the “Permit” input is set to TRUE, controller commands can be processed. If the “Permit” input is set to FALSE, control commands are no longer processed. Control commands (e.g. for valves) are no longer executed. No control outputs are set either (e.g. at the operators). The FB retains the most recent state – only the safety position/value can be approached. The FB then remains in this position until the “Permit” input is set back to TRUE.

Note:

If the safe position is set to “running”, switching between forward and reverse running is possible via “Permit”. Only stopping the motor is not possible. If this type of performance is not desired, one of the directions can be deactivated via “FwdEn” and “RevEn” respectively. Rotational speed can also be varied.

14.1.3.8.2 Interlock

The interlock function can be activated via the “IntlEn” input (TRUE). If the “Interlock” input is set to TRUE, control commands can be processed. If the “Interlock” input is set to FALSE, no control commands are processed. The safety position/value is approached, if the FB has one. If not, the most recent state is retained. No control commands are processed until the “Interlock” input is set back to TRUE or the function is deactivated via “IntlEn”. When the “Interlock” input is once again set to TRUE, the FB remains in the safety state until a new control command is transmitted.

If the FB is in the safety state, “SafePosAct” is set to TRUE.

14.1.3.8.3 Protection lock

The protection lock function can be activated via the “ProtEn” input (TRUE). If the “Protect” input is set to TRUE, control commands can be processed. If the “Protect” input is set to FALSE, no control commands are processed. The safety position/value is approached, if the FB has one. If not, the most recent state is retained. Like the interlock function, no control commands are processed until the “Protect” input is set back to TRUE or the function is deactivated via “ProtEn”. The protection lock must be additionally reset. This is possible in the visualisation via the faceplate (“ResetOp”) or via the “ResetAut” input at the FB.

Note:

- The lock function is active in all states (offline, operator, automatic).
- “ResetAut” is edge-triggered and reacts to rising edges.

14.1.4 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
40d	16#28	2	Bad, conditioned by process	Offline mode it active
				$Rpm < RpmMin$ ⇒ $Rpm = RpmMin$
				$Rpm > RpmMax$ ⇒ $Rpm = RpmMax$
				$RpmMin > RpmMax$
				$RpmSclMax < RpmSclMin$ ⇒ $RpmSclMax = RpmMax$ ⇒ $RpmSclMin = RpmMin$
				$RpmSclMax < RpmMin$ or $RpmSclMax > RpmMax$ ⇒ $RpmSclMax = RpmMax$
				$RpmSclMin < RpmMin$ or $RpmSclMin > RpmSclMax$ ⇒ $RpmSclMin = RpmMin$
				$RpmSclMax - RpmSclMin = 0$ ⇒ $RpmSclMax = RpmMax$ ⇒ $RpmSclMin = RpmMin$
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 83 AnaDrv specific worst quality code

14.1.5 Source mode

14.1.5.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (manual). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (*Int) the SrcIntAct output is set to TRUE. When the manual source is active (*Man), the SrcManAct output bit is set to TRUE.

The internal source is active as a default setting.

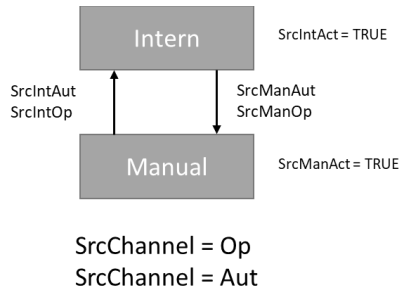


Fig. 88 Statemachine source mode

14.1.5.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutoOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcManAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAut” and “SrcManAut” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcManAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcManAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

14.1.5.3 Modul-external access (OPC UA)

The “SrcIntOp” and “SrcManOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutoOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “SrcIntOp” (internal source) or “SrcManOp” (manual source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcManOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcManAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn't necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they're acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntOp” and “SrcManOp”) are set simultaneously, “SrcIntOp” has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “SrcIntOp”, “SrcManOp”, “SrcIntAut” or “SrcManAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it's important to ensure that these inputs are not permanently assigned to the FB.

14.1.6 Operation mode

14.1.6.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode. Changes of state are only possible between offline and operator or operator and automatic. Changing directly from offline to automatic is not allowed.

The “StateChannel” input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the “StateOffAct” output is set to TRUE. When the operator mode is active “StateOpAct” is set to TRUE, and when the automatic mode is active “StateAutAct” is set to TRUE.

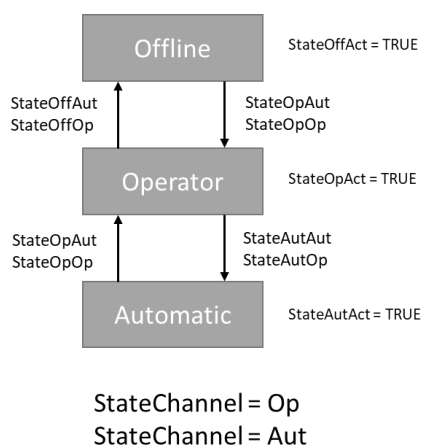


Fig. 89 Statemachine operation mode

14.1.6.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- “StateChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “StateOffAut” input (switch to offline mode), the “StateOpAut” input (switch to operator mode) or the “StateAutAut” input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

14.1.6.3 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.
- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the

variable is set back to FALSE in order to acknowledge the request. For this reason, it's important to ensure that these inputs are not permanently assigned to the FB.

14.1.7 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object. The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
SafePos	BOOL	FALSE	Safe Position (1: running 0: stopped)
OSLevel	BYTE	0	OS Level Variable

Tab. 84 AnaDrv persistent variables

14.1.8 Symbols

14.1.8.1 Description

Element	Description	Rule
Tagname	Name of the unit of measure – indicated in the visualisation.	Max. 16 characters.
Drive symbol	Grey: motor is stopped Blue: motor is running	
Operation Mode	 : offline mode is active  : operator mode is active  : automatic mode is active	See description of operation mode.
Interlock	: no interlock or protection lock is active  : interlock is active  : protection lock is active	See lock function description.
Permission	 : access permitted  : access denied	See lock function description.
Drehrichtung	 : motor running forward  : motor running in reverse	No symbol is displayed when the motor is stopped.
Rpm	Variable “RpmFbk” is displayed.	Max. 5 places before and 2 places after the decimal point are displayed.
Unit	“RpmUnit” is displayed.	Only values from the “UnitSettings” enumeration are possible.
Nozzles	Nozzles for connecting logical lines (dashed nozzles), measurement lines (nozzle with solid line) and pipes (blue nozzles).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 85 AnaDrv symbol description

14.1.8.2 dAnaDrvPumpValue/uAnaDrvPumpValue/rAnaDrvPumpValue/lAnaDrvPumpValue

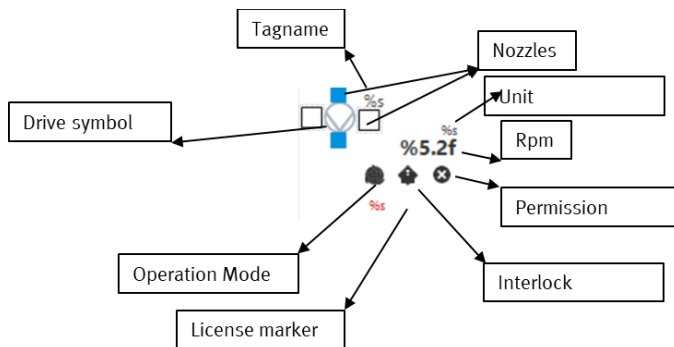


Fig. 90 Symbol AnaDrv Pump

14.1.8.3 hAnaDrvMotorValue

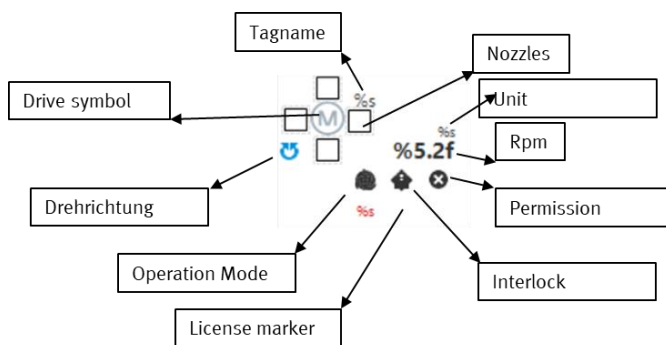


Fig. 91 Symbol hAnaDrvMotorValue

14.2 MonAnaDrv function block

14.2.1 Brief description

The MonAnaDrv function block (FB) expands the AnaDrv FB to include feedback monitoring for dynamic and static errors, as well as speed monitoring.

14.2.2 Extended functions

14.2.2.1 Activate feedback monitoring

Feedback monitoring (static and dynamic) can be activated via the “MonEn” input at the FB (= TRUE). If “MonEn” is set to FALSE or if it’s set to FALSE during operation, all error outputs are reset. The alarm symbol also disappears from the visualisation and the error doesn’t need to be acknowledged.

14.2.2.2 “MonSafePos” input

The “MonSafePos” input at the FB can be used to specify whether the motor enters the safety state in the event of an error (MonSafePos = TRUE) or the current state is retained (MonSafePos = FALSE). In either case, the “SafePosAct” output bit is set to TRUE in order to detect the exception state. As a default setting, travel to the safety position takes place in the event of an error.

If the most recent state should be retained after an error, or if the motor is already in the safety state, errors are not eliminated until corresponding feedback signals provide the correct values.

If the motor is set to the safety state after an error, the error is not eliminated until after returning to the state in which the error occurred and the feedback signals provide the correct values.

Example:

The motor is stopped and needs to be started. After starting, “FwdFbk” is still set to FALSE. The motor is thus set to its safety state again (in this case stopped). In order to eliminate the error, the motor must be started again and “FwdFbk” must be set to TRUE. Only then are the error outputs (“StatErrAct” and “DynErrAct”) set to FALSE.

14.2.2.3 Static errors

A static error has occurred when the feedback signal changes although no change has taken place at the controller.

The following static errors are possible:

- “FwdFbk” and “RevFbk” are both TRUE.
- State is running, “FwdFbk”/ “RevFbk” is switched to FALSE.
- State is stopped, “FwdFbk”/ “RevFbk” is switched to TRUE.

In order to assure that a malfunction is not assessed as a lost or incorrect signal, the error state must persist for a certain duration in order for it to be evaluated as an error. This duration can be set via the “MonStatTi” input at the FB. The error is not recognised and the “MonStatErr” output is not set to TRUE unless the error state is still pending after this time has expired.

If a visualisation is used in CODESYS, the corresponding alarm appears in the symbol.

14.2.2.4 Dynamic errors

A dynamic error has occurred when the feedback signal doesn't change although a change has taken place at the controller.

The following dynamic errors are possible:

- State should change from running to stopped – “FwdFbk” remains TRUE
- State should change from stopped to running – “FwdFbk” remains FALSE
- Stat should change from forward running to reverse running – “RevFbk” remains FALSE

In order to assure that the feedback signal is not expected immediately (because a motor requires a certain amount of time to start up or stop), a duration can be set via the “MonDynTi” input at the FB. If the correct feedback signal is not detected within this period of time, the “DynErrAct” output is set to TRUE.

If a visualisation is used in CODESYS, the corresponding alarm appears in the symbol.

14.2.2.5 Speed monitoring

This function is used to monitor speed “RpmFbk”. A lower limit value “RpmALLim” and an upper limit value “RpmAHLim” can be entered, which speed “RpmFbk” may not fall short of or exceed respectively.

The limit values are available as inputs at the FB and can be activated separately via the “RpmALEn” and “RpmAHEn” inputs.

Beyond this, a time can be specified at the FB via the “MonRpmTi” input. If a limit value violation is detected, it must persist for a duration of at least “MonRpmTi” in order to be recognised as an error. Otherwise, it's considered a signal flutter and no alarm is triggered.

An alarm is read out via the associated output bit (“RpmALerr” and “RpmAHerr”), but there's no travel to a safety position. The difference between “Rpm” and “RpmFbk” is read out via the “RpmErr” output.

If a visualisation is used in CODESYS, the corresponding alarm appears in the symbol.

If limit values are set incorrectly (e.g. lower limit value > upper limit value), the WQC is set correspondingly.

Note:

- If a motor is stopped, “RpmFbk” is not checked against the limit value, and instead it's determined whether the value is less than 0.5.

14.2.2.6 Alarm priority

The following priorities are adhered to for the alarm display in the visualisation:

- Both feedbacks active (top priority)
- Forward feedback lost
- Reverse feedback lost
- Stop feedback lost
- Failed to run forward
- Failed to run in reverse
- Failed to stop
- Rpm high alarm active
- RpmLow alarm active
- No alarm

14.2.2.7 Time “MonStatTi”, “MonDynTi”, “MonRpmTi”

The “MonStatTi”, “MonDynTi” and “MonRpmTi” variables are specified in seconds and are set to 0.5 s as a default value. This duration can be changed via the input at the FB or at the faceplate. Values of less than 0 are converted to positive numbers. For example, if -5 s is entered, time is set to 5 s.

14.2.2.8 Acknowledging alarms

If no errors are pending anymore, the corresponding output bits are set back to FALSE. However, the error in the symbol must be reset by operator via the faceplate (“MonAckOp”) or via the “MonAckAut” input at the FB. Visibility of the alarm symbol is described in the table.

Limit value violation still pending	Error acknowledged	Symbol
Yes	No	Blinking
Yes	Yes	Visible
No	No	Blinking
No	Yes	Invisible – no error

Tab. 86 Alarm symbol states

Note:

- The “MonAckAut” input is edge-triggered and only reacts to rising edges.
- When acknowledged, all currently pending alarms are acknowledged simultaneously.
- Multiple limit errors may also be pending – only the highest priority is displayed in the visualisation.

14.2.3 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
40d	16#28	2	Bad, conditioned by process	Offline mode it active
				$Rpm < RpmMin$ ⇒ $Rpm = RpmMin$
				$Rpm > RpmMax$ ⇒ $Rpm = RpmMax$
				$RpmMin > RpmMax$
				$RpmSclMax < RpmSclMin$ ⇒ $RpmSclMax = RpmMax$ ⇒ $RpmSclMin = RpmMin$
				$RpmSclMax < RpmMin$ or $RpmSclMax > RpmMax$ ⇒ $RpmSclMax = RpmMax$
				$RpmSclMin < RpmMin$ or $RpmSclMin > RpmSclMax$ ⇒ $RpmSclMin = RpmMin$
				$RpmSclMax - RpmSclMin = 0$ ⇒ $RpmSclMax = RpmMax$ ⇒ $RpmSclMin = RpmMin$
				Limit values $> RpmMax$ and/or limit value $< RpmMin$ (only when they're activated)
				Upper limit value $<$ lower limit value
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 87 MonAnaDrv specific worst quality code

14.2.4 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
SafePos	BOOL	FALSE	Safe Position (1: running 0: stopped)
OSLevel	BYTE	0	OS Level Variable
MonEn	BOOL	FALSE	Enables dynamic and static Feedback Monitoring (1: enabled 0: disabled)
MonSafePos	BOOL	TRUE	Enables Monitor Safe Position (1: go to Safe Position 0: Hold last state)
MonStatTi	REAL	0.5	Monitor Time for uncontrolled changes [s]
MonDynTi	REAL	0.5	Monitor Time for controlled changes [s]
RpmAHEn	BOOL	FALSE	Enables Alarm High Limit (1: enabled 0: disabled)
RpmAHLim	REAL	0	Limit Value for Alarm High
RpmALEn	BOOL	FALSE	Enables Alarm Low Limit (1: enabled 0: disabled)
RpmALLim	REAL	100	Limit Value for Alarm Low
MonRpmTi	REAL	0.5	Monitor Time to recognize an Rpm limit error [s], only positive values

Tab. 88 MonAnaDrv persistent variables

14.2.5 Symbols

14.2.5.1 Description

Element	Description	Rule
Tagname	Name of the unit of measure – indicated in the visualisation.	Max. 16 characters.
Alarm symbols	 : alarm is active	See description of function block and alarm function.
Drive symbol	Grey: motor is stopped Blue: motor is running	
Operation Mode	 : offline mode is active  : operator mode is active  : automatic mode is active	See description of operation mode.
Interlock	 : no interlock or protection lock is active  : interlock is active  : protection lock is active	See lock function description.
Permission	 : access permitted  : access denied	See lock function description.
Drehrichtung	 : motor running forward  : motor running in reverse	No symbol is displayed when the motor is stopped.
Rpm	Variable “RpmFbk” is displayed.	Max. 5 places before and 2 places after the decimal point are displayed.
Unit	“RpmUnit” is displayed.	Only values from the “UnitSettings” enumeration are possible.
Nozzles	Nozzles for connecting logical lines (dashed nozzles), measurement lines (nozzle with solid line) and pipes (blue nozzles).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.

Element	Description	Rule
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 89 MonAnaDrv symbol description

14.2.5.2 dMonAnaDrvPumpValue/uMonAnaDrvPumpValue/rMonAnaDrvPumpValue/lMonAnaDrvPumpValue

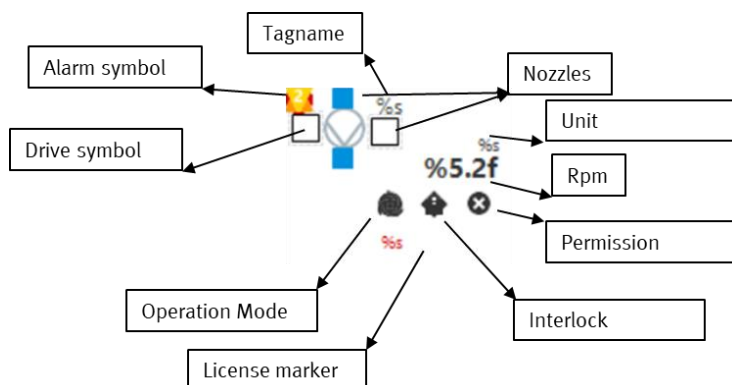


Fig. 92 Symbol MonAnaDrv Pump

14.2.5.3 hMonAnaDrvValue

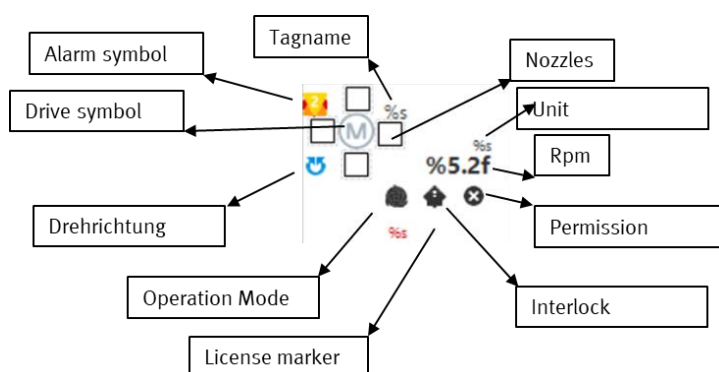


Fig. 93 Symbol hMonAnaDrvValue

14.2.6 Faceplates

14.2.6.1 Main

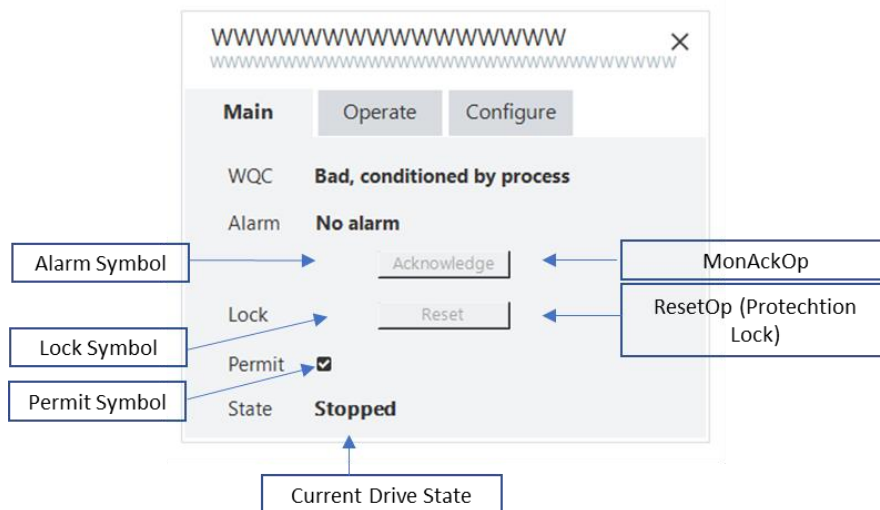


Fig. 94 MonAnaDrv main faceplate

14.2.6.2 Operate

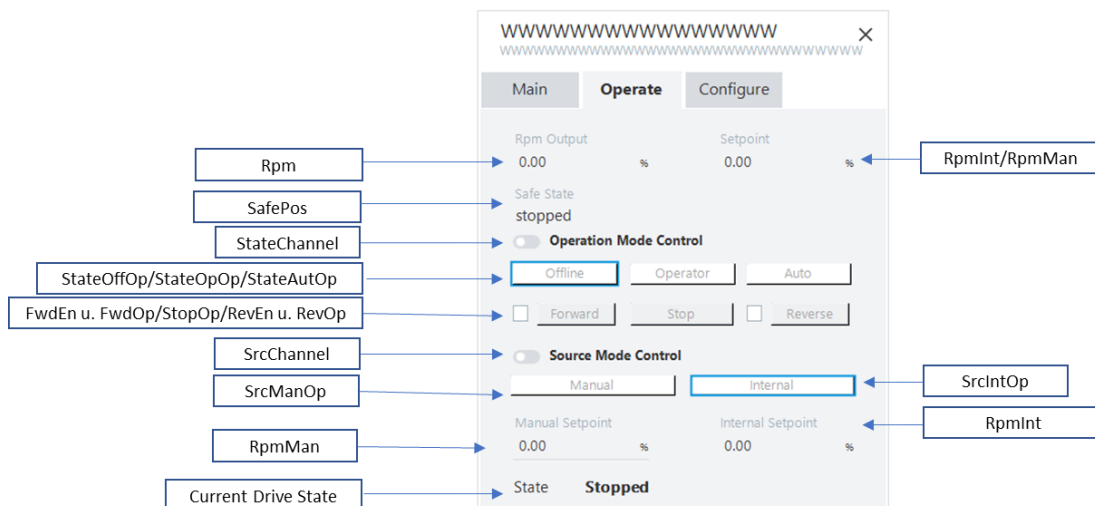


Fig. 95 MonAnaDrv operate faceplate

14.2.6.3 Configure

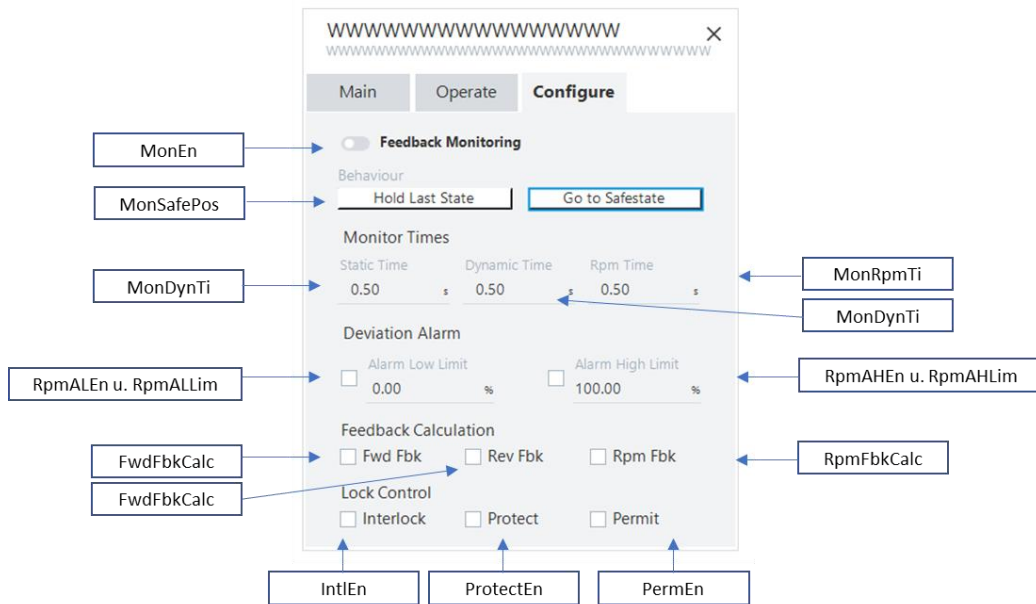


Fig. 96 MonAnaDrv configure faceplate

15 Totalizer

15.1 Totalizer function block

15.1.1 Brief description

The totalizer function block (FB) makes it possible to implement a dosing process.

15.1.2 Basic functions

15.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

15.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 90 Worst quality code enumeration

15.1.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

15.1.3 Functions

15.1.3.1 Setpoint specification for “Splnt” and “SpMan”

The setpoint for the quantity to be proportioned can be specified via two sources. The setpoint can be specified directly at the FB via the “Splnt” input. If the setpoint will be selected via OPC UA or the faceplate, the entry field „Manual Setpoint“ is used. The source mode controls which of the two inputs will be used for the dosing process at any time. Upon initialisation, the internal source is active, i.e. “Splnt”. The mode of operation and handling of the source mode is explained in a separate section.

15.1.3.2 Setpoint limiting

Limit values can be entered for the setpoint value via the “SpMin” and “SpMax” inputs at the FB. If the specified setpoint (“Splnt” or “SpMan”) is greater than “SpMax” or less than “SpMin”, the setpoint is limited to the corresponding limit value. It must also be assured that “SpMin” is not greater than “SpMax”. Otherwise, there’s no guarantee that the conversion and the graphics in the visualisation will include sensible values.

15.1.3.3 Unit of measure for “FlowIn” and “TotalDos”

A unit of measure can be entered for flow rate “FlowIn” via the “FlowUnit” input at the FB. A unit of measure can be entered for the dosed quantity “TotalDos” via the “TotalDosUnit” input at the FB. The “UnitSettings” enumeration is used in order to specify a unit of measure. The unit of measure is only used for displays and has no influence on the conversion.

15.1.3.4 Calculating flow rate “FlowIn”

The “FlowIn” input is linked to the measured flow rate. For example, this can come directly from a flow sensor. A time basis for “FlowIn” is required in order to calculate the current flow rate. This value is specified via the “FlowScl” input at the FB in seconds and is set to 60 s as a default value. If a value of less than 0 is entered, 60 s is used for calculation. The already dosed quantity is read out via the “TotalDos” output.

15.1.3.5 “InFlight” input parameter

When the valve will closed can be adjusted via the “InFlight” input at the FB. The point in time at which the dose quantity reaches the value “Splnt/SpMan – InFlight” is determined to this end. And thus the valve closes before the actual setpoint value is reached. This prevents exceeding the setpoint value in the case of large valves. If values of less than 0 are entered, a value of 0 is used for internal calculation.

States of the InFlight function:

Conditions	Operation mode	Effect
TotalDos > Sp- InFlight	Manual/Auto	Valve state = 1
TotalDos = Sp - InFlight	Manual/Auto	Valve state = 0
Setpoint < InFlight	Manual/Auto	InFlight = 0

Tab. 91 States of InFlight

15.1.3.6 Cut-off

Due to the fact that the “FlowIn” value is never exactly 0 in the case of a closed valve, the “zero value” can be entered via the “CutOff” input at the FB. If “FlowIn” is less than “CutOff”, the valve is deemed closed. This means that after a start command only quantities of greater than “CutOff” at the “FlowIn” input are seen as flow and used for calculation. If values of less than 0 are entered, a value of 0 is used for internal calculation.

States of the CutOff function:

Conditions	Operation mode	Effect
CutOff < 0	Manual/Auto	Cutoff = 0
CutOff > FlowIn	Manual/Auto	Valve state = 0
CutOff < FlowIn + SP > 0	Manual/Auto	Valve state = 1

Tab. 92 States of CutOff

15.1.3.7 Totalizer control

The totalizer can be controlled via the start, stop and reset control commands. The start command opens the valve and starts the dosing process. “FlowIn” must be greater than “CutOff” in order for the value to be taken into consideration. Dosing continues until the setpoint has been reached (in consideration of InFlight). The valve is then closed. When “FlowIn” is less than “CutOff”, the valve is deemed closed and the totalizer enters the “Finished” state.

When the stop control command is issued, the valve is closed and the filling process is paused. When the reset command is issued, the filling process is stopped and all values are reset, so that a new dosing process can be started.

Depending on the command, control commands are issued by setting the above-mentioned variables to TRUE. Input variables “StartAut”, “StopAut” and “ResetAut” are edge-triggered and react to rising edges. “StartOp”, “StopOp” and “ResetOp” are set back to FALSE after registration by the FB.

The following table shows how the VlvOut output performs depending on the safety position and the control commands.

Control command	Outputs
Stop	VlvOut = 0 (valve closed)
Start	VlvOut = 1
Reset	VlvOut = 0

Tab. 93 Control commands

The FB can be operated in three states, namely offline, operator and automatic. The states are controlled via the operation mode. This mode is described in a separate section.

The following control commands and outputs are processed and set in the respective state:

- Offline mode: no reset function is executed, VlvOut = 0
- Operator mode: “StartOp”, “StopOp”, “ResetOp”, “SPMan” or “SPInt” (can be changed via faceplate)
- Automatic mode: “StartAut”, “StopAut”, “ResetAut”, “SPMan” or “SPInt” (inputs at FB)

15.1.4 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
40d	16#28	2	Bad, conditioned by process	Offline mode it active
				FlowScl ≤ 0 ⇒ FlowScl = 60
				InFlight < 0 ⇒ InFlight = 0
				InFlight > Sp ⇒ InFlight = 0
				CutOff < 0 ⇒ CutOff = 0
				TotalDos < 0
				SpMin > SPMax
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 94 Totalizer specific worst quality code

15.1.5 Source mode

15.1.5.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (manual). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (*Int) the SrcIntAct output is set to TRUE. When the manual source is active (*Man), the SrcManAct output bit is set to TRUE.

The internal source is active as a default setting.

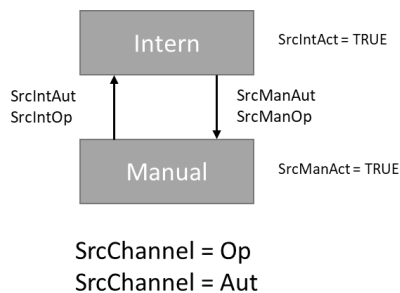


Fig. 97 Statemachine source mode

15.1.5.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcManAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAut” and “SrcManAut” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcManAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcManAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

15.1.5.3 Modul-external access (OPC UA)

The “SrcIntOp” and “SrcManOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “SrcIntOp” (internal source) or “SrcManOp” (manual source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcManOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcManAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntOp” and “SrcManOp”) are set simultaneously, “SrcIntOp” has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “SrcIntOp”, “SrcManOp”, “SrcIntAut” or “SrcManAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

15.1.6 Operation mode

15.1.6.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode. Changes of state are only possible between offline and operator or operator and automatic. Changing directly from offline to automatic is not allowed.

The “StateChannel” input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the “StateOffAct” output is set to TRUE. When the operator mode is active “StateOpAct” is set to TRUE, and when the automatic mode is active “StateAutAct” is set to TRUE.

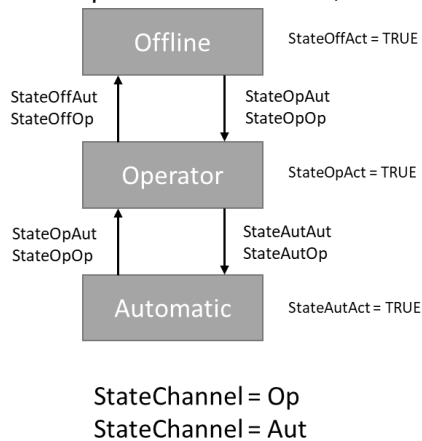


Fig. 98 Statemachine operation mode

15.1.6.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- “StateChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “StateOffAut” input (switch to offline mode), the “StateOpAut” input (switch to operator mode) or the “StateAutAut” input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

15.1.6.3 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.
- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

15.1.7 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
InFlight	REAL	0	Parameter to stop the dosing process before setpoint is reached (SpInt/SpMan - InFlight), only positive values
CutOff	REAL	0	No Flow Parameter (set the zero value for no Flow Input), only positive values
OSLevel	BYTE	0	OS Level Variable

Tab. 95 Totalizer persistent variables

15.1.8 Symbols

15.1.8.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Operation Mode	 : offline mode is active  : operator mode is active  : automatic mode is active	See description of operation mode.
Current setpoint	Depending on the source mode, “SpInt” or “SpMan” is displayed.	Max. 5 places before and 2 places after the decimal point are displayed.
Dosed amount	“TotalDos” is displayed.	Max. 5 places before and 2 places after the decimal point are displayed.
Unit	U/M “TotalDosUnit” is displayed for the setpoint value and the dosed quantity.	Only values from the “UnitSettings” enumeration are possible.
Nozzles	Nozzles for connecting logical lines (dashed line).	
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 96 Totalizer symbol description

15.1.8.2 hTotalizerValue

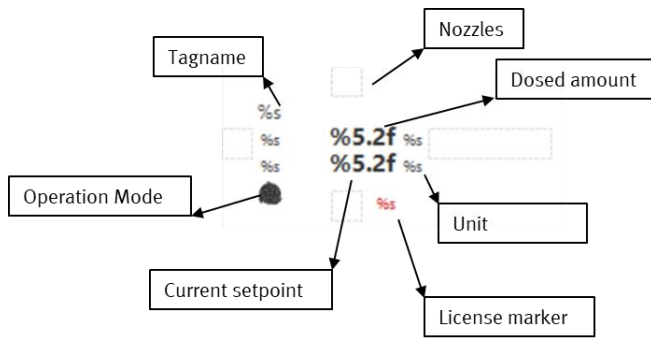


Fig. 99 Symbol hTotalizerValue

15.2 MonTotalizer function block

15.2.1 Brief description

The MonTotalizer function block (FB) expands the Totalizer FB to include limit value monitoring.

15.2.2 Extended Functions

15.2.2.1 Check dosed quantity “TotalDos“

This function checks dosed quantity “TotalDos” at the end of a dosing process, i.e. when the “Finished” state has been reached (setpoint reached, valve closed and FlowIn < CutOff). Lower (“TolALLim”) and upper (“TolAHLim”) tolerance values can be entered as percentages via inputs at the FB. A tolerance range is calculated using these percentage values and the setpoint. After the dosing process has been ended, checking takes place to determine whether or not the “TotalDos” output is within the calculated tolerance range. If not an alarm is read out. The tolerance values can be activated individually via the inputs at the FB (“TolALEn” and “TolAHEn”).

If values of less than 0 are entered as tolerance values, these are converted to positive numbers. For example, if -5% is entered, a value of 5% is used.

An alarm is read out via the associated output bit (“TolALerr” and “TolAHerr”).
If a visualisation is used in CODESYS, the corresponding alarm appears in the symbol.

15.2.2.2 Acknowledging alarms

If no errors are pending anymore, the corresponding output bits are set back to FALSE. However, the error in the symbol must be reset by operator via the faceplate (“TolAlmAckOp”) or via the “TolAlmAckAut” input at the FB. Visibility of the alarm symbol is described in the table.

Limit value violation still pending	Error acknowledged	Symbol
Yes	No	Blinking
Yes	Yes	Visible
No	No	Blinking
No	Yes	Invisible – no error

Tab. 97 Alarm symbol states

Note:

- The “TolAlmAckAut” input is edge-triggered and only reacts to rising edges.

15.2.3 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
40d	16#28	2	Bad, conditioned by process	Offline mode it active
				FlowScl ≤ 0 ⇒ FlowScl = 60
				InFlight < 0 ⇒ InFlight = 0
				InFlight > Sp ⇒ InFlight = 0
				CutOff < 0 ⇒ CutOff = 0
				TotalDos < 0
				SpMin > SPMax
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 98 MonTotalizer specific worst quality code

15.2.4 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
InFlight	REAL	0	Parameter to stop the dosing process bevor setpoint is reached (Splnt/SpMan - InFlight), only positive values
CutOff	REAL	0	No Flow Parameter (set the zero value for no Flow Input), only positive values
OSLevel	BYTE	0	OS Level Variable
TotAHEn	BOOL	FALSE	Enables Alarm High Limit (1: enabled 0: disabled)
TotAHLim	REAL	0	Limit Value for Alarm High [%]
TotALEn	BOOL	FALSE	Enables Alarm Low Limit (1: enabled 0: disabled)
TotALLim	REAL	0	Limit Value for Alarm Low [%]

Tab. 99 MonTotalizer persistent variables

15.2.5 Symbols

15.2.5.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Alarm symbols	 : alarm is active (top priority)	See description of function block and alarm function.
Operation Mode	 : offline mode is active  : operator mode is active  : automatic mode is active	See description of operation mode.
Current setpoint	Depending on the source mode, “Splnt” or “SpMan” is displayed.	Max. 5 places before and 2 places after the decimal point are displayed.
Dosed amount	“TotalDos” is displayed.	Max. 5 places before and 2 places after the decimal point are displayed.
Unit	U/M “TotalDosUnit” is displayed for the setpoint value and the dosed quantity.	Only values from the “UnitSettings” enumeration are possible.
Nozzles	Nozzles for connecting logical lines (dashed line).	Important for the MTP exporter in order to assure that lines are connected correctly to the symbol and to other lines.
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 100 MonTotalizer symbol description

15.2.5.2 hMonTotalizerValue

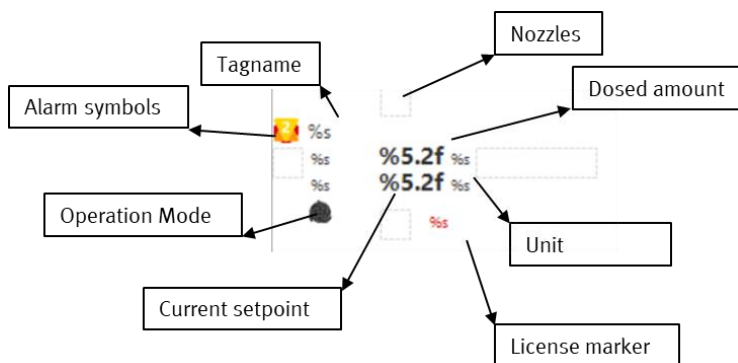


Fig. 100 Symbol hMonTotalizerValue

15.2.6 Faceplates

15.2.6.1 Main

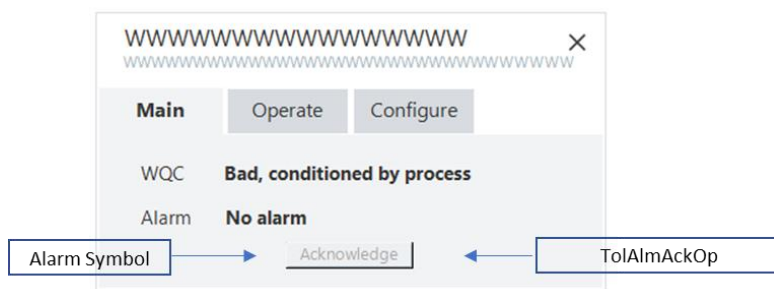


Fig. 101 MonTotalizer main faceplate

15.2.6.2 Operate

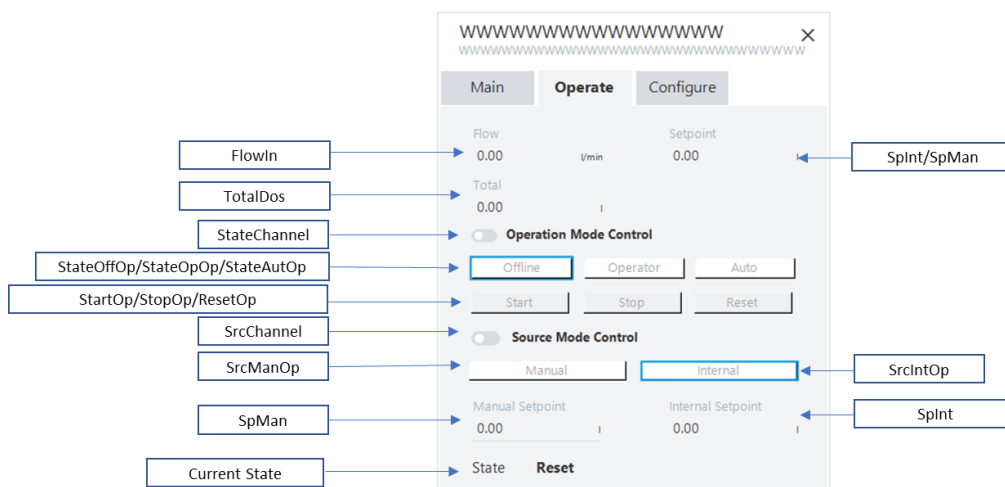


Fig. 102 MonTotalizer operate faceplate

15.2.6.3 Configure

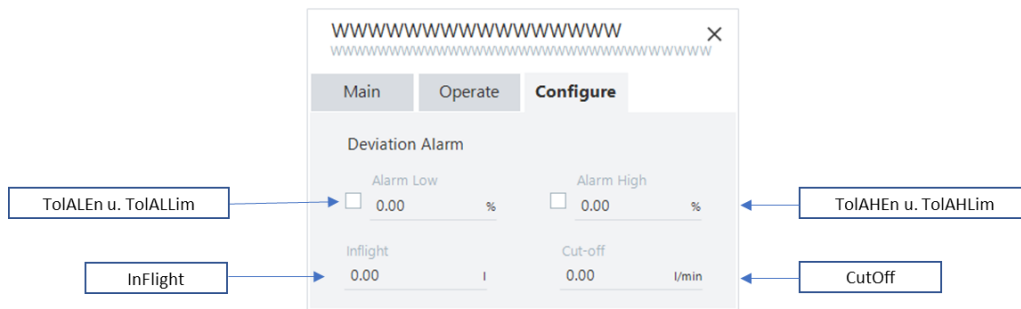


Fig. 103 MonTotalizer configure faceplate

16 MPA positioner

16.1 DrvCtrl function block

16.1.1 Brief description

The DrvCtrl function block (FB) provides the option of operating a rotary or linear drive via pilot valves, and of sensing its position. Together with the valves and analogue position feedback, the function of a positioner is fulfilled.

Depending on the size of the drive, desired drive speed and positioning accuracy, a coarse valve can be controlled in addition to the fine valve.

The MPA positioner cannot be used until calibration has been conducted. The drive is advanced to its end positions to this end, and the respective position feedback is saved. These values constitute the drive's scale from 0% to 100%.

16.1.2 Basic functions

16.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

16.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 101 Worst quality code enumeration

16.1.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

16.1.3 Functions

16.1.3.1 Setpoint specification for “SpMan” and “Splnt”

The setpoint for the position can be specified via two sources. The setpoint can be specified directly at the FB via the “Splnt” input. If the setpoint will be selected via OPC UA or the faceplate, the entry field „Manual Setpoint“ is used. The source mode controls which of the two inputs will be used for positioning at any time. Upon initialisation, the internal source is active, i.e. “Splnt”. The mode of operation and handling of the source mode is explained in a separate section.

16.1.3.2 Setpoint limiting

The limitation of the setpoint is not configurable and takes place via the internal variables “SpMin” and “SpMax”. “SpMin” has a predefined value of 0 and “SpMax” has a predefined value of 100. If the specified setpoint (“Splnt” or “SpMan”) is greater than “SpMax” or less than “SpMin”, the setpoint is limited to the corresponding limit value.

16.1.3.3 Process value feedback signals (simulated, actual PV)

The actual process of value (PV) can be linked via the “FV” input at the am FB. This is then converted to the actual “PV”. The PV can also be simulated via the inputs at the “PVSimType” FB by specifying a value. It's also possible to set the PV to the current setpoint value.

16.1.3.4 Units of measure

As is commonly the case for positions, the unit of measure is % (percent).

16.1.3.5 Safety position – FailSafe

In the event of an error or locking (e.g. interlock) The “FailSafeVlv” output is set to FALSE. Furthermore, all outputs (GoTo0Vlv, GoTo100Vlv, CoarseGoTo0Vlv, Coarse GoTo100Vlv) are set to a value of 0.

16.1.3.6 Positioner function

The FB can be operated in three states, namely offline, operator and automatic. The states are controlled via the operation mode. This mode is described in a separate section.

The following control commands and outputs are processed and set in the respective state:

- Offline mode: GoTo0Vlv = 0, GoTo100Vlv = 0, CoarseGoTo0Vlv = 0, Coarse GoTo100Vlv = 0, FailSafeVlv = 0
- Operator mode: Move to 0% and Move to 100% (= TRUE) traverse until the input is FALSE again (corresponding valves are actuated).
- Automatic mode: valves are controlled depending on PV and setpoint value

16.1.3.7 “UseCoarseVlv” input parameter

The function for “Coarse Valve (CV)” can be deactivated in the faceplate via this input.

16.1.3.8 Calibration

Calibration can be started in the faceplate via the "Start" button in the "Calibrate" tab or directly at the FB via “CalibrateExt” by setting the respective variable to TRUE.

16.1.3.9 Lock functions

16.1.3.9.1 Permit

The permit function can be activated via the “PermEn” input (= TRUE). If the “Permit” input is set to TRUE, controller commands can be processed. IF the “Permit” input is set to FALSE, control commands are no longer processed. Control commands (e.g. for valves) are no longer executed. No control outputs are set either (e.g. at the operators). The FB retains the most recent state – only the safety position can be approached. The FB then remains in this position until the “Permit” input is set back to TRUE.

16.1.3.9.2 Interlock

The interlock function can be activated via the “IntlEn” input (TRUE). If the “Interlock” input is set to TRUE, control commands can be processed. IF the “Interlock” input is set to FALSE, no control commands are processed. The safety position is approached, if the FB has one. If not, the most recent state is retained. No control commands are processed until the “Interlock” input is set back to TRUE or the function is deactivated via “IntlEn”. When the “Interlock” input is once again set to TRUE, the FB remains in the safety state until a new control command is transmitted.

16.1.3.9.3 Protection lock

The protection lock function can be activated via the “ProtEn” input (TRUE). If the “Protect” input is set to TRUE, control commands can be processed. IF the “Protect” input is set to FALSE, no control commands are processed. The safety position is approached, if the FB has one. If not, the most recent state is retained. Like the interlock function, no control commands are processed until the “Protect” input is set back to TRUE or the function is deactivated via “ProtEn”. The protection lock must be additionally reset. This is possible in the visualisation via the “Reset” button in the "Main" tab or via the “ResetAut” input at the FB.

Note:

- The lock function is active in all states (offline, operator, automatic).
- “ResetAut” is edge-triggered and reacts to rising edges.

16.1.3.9.4 Position monitoring

The dead band can be set via the “DeadBand” input at the FB. This is added to and subtracted from the current setpoint. This results in a tolerance range. When the positioner is set into motion the setpoint’s tolerance range must be reached within duration “AlmDevTi” – Otherwise, an error is read out. An alarm is read out via the associated “AlmDevAct” output bit. If a visualisation is used in CODESYS, the corresponding alarm appears in the symbol.

Numbers for the “DeadBand” parameter must be positive. If members of less than 0 are specified, they’re converted to positive numbers. For example, if -5 is entered, time is set to 5 s.

Positioning monitoring can be activated via the “AlmDevEn” input at the FB (= TRUE).

16.1.3.10 Time “AlmDevTi”

The “AlmDevTi” variable is specified in seconds and is set to 0.5 s as a default value. This duration can be changed via the input at the FB or at the faceplate via the entry field “Alarm Time”. Values of less than 0 are converted to positive numbers. For example, if -5 s is entered, time is set to 5 s.

16.1.3.11 Acknowledging alarms

If no limit value violations are pending anymore, the corresponding output bits “AlmDevAct” are set back to FALSE. However, the error in the symbol must be reset by operator via the faceplate (“Acknowledge”) or via the “AlmDevAckAut” input at the FB. Visibility of the alarm symbol is described in the table.

Limit value violation still pending	Error acknowledged	Alarm Symbol
Yes	No	Blinking
Yes	Yes	Visible
No	No	Blinking
No	Yes	Invisible – no error

Tab. 102 Alarm symbol states

Note:

- The “AlmDevAckAut” input is edge-triggered and only reacts to rising edges.
- When acknowledged, all currently pending alarms are acknowledged simultaneously.
- Multiple limit value violations may also be pending and can be read out via the corresponding output bits at the same time – only the highest priority is displayed in the visualisation.

16.1.4 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
40d	16#28	2	Bad, conditioned by process	Offline mode is active
				SpInt/SpMan < SpMin ⇒ Sp = SpMin
				SpInt/SpMan > SpMax ⇒ Sp = SpMax
				SpMin > SpMax
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 103 DrvCtrl specific worst quality code

16.1.5 Source mode

16.1.5.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (manual). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (*Int) the SrcIntAct output is set to TRUE. When the manual source is active (*Man), the SrcManAct output bit is set to TRUE.

The internal source is active as a default setting.

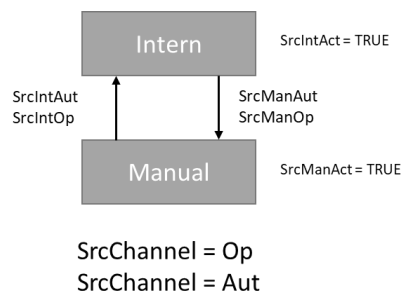


Fig. 104 Statemachine source mode

16.1.5.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutoOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcManAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAut” and “SrcManAut” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcManAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcManAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

16.1.5.3 Module-external access (OPC UA)

The “SrcIntOp” and “SrcManOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “SrcIntOp” (internal source) or “SrcManOp” (manual source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcManOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcManAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntOp” and “SrcManOp”) are set simultaneously, “SrcIntOp” has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “SrcIntOp”, “SrcManOp”, “SrcIntAut” or “SrcManAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

16.1.6 Operation mode

16.1.6.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode. Changes of state are only possible between offline and operator or operator and automatic. Changing directly from offline to automatic is not allowed.

The “StateChannel” input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the “StateOffAct” output is set to TRUE. When the operator mode is active “StateOpAct” is set to TRUE, and when the automatic mode is active “StateAutAct” is set to TRUE.

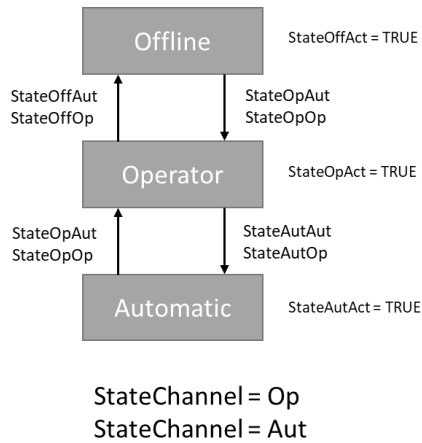


Fig. 105 Statemachine operation mode

16.1.6.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- “StateChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “StateOffAut” input (switch to offline mode), the “StateOpAut” input (switch to operator mode) or the “StateAutAut” input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

16.1.6.3 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.
- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

16.1.7 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It’s even possible to make a selection specifying which variables will be persistently stored.

- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
Scale100	REAL	20000	High Scale Value of Input (Scaling factor for PV calculation)
Scale0	REAL	4000	Low Scale Value of Field Value (Scaling factor for PV calculation)
AlmDevEn	BOOL	FALSE	Enables Deviation Alarm (1: enabled 0: disabled)
DeadBand	REAL	1.0	Deadband Limit Value (Inside the deadband the output is tracked, tolerance value for deviation alarm)
AlmDevTi	REAL	0.5	Monitor Time for Deviation Alarm [s]
Calibrated	BOOL	FALSE	Calibration has been finished
OSLevel	BYTE	0	OS Level Variable
CoarseThreshold	REAL	10.0	Full power with coarse valve over +ThresholdCoarse and under - ThresholdCoarse [%]
Threshold	REAL	5.0	Begin to pulse under +Threshold and above - Threshold [%]
PWMFreq	REAL	7.5	Sets the period for the pulsing sequence: 10 means 100ms
CyclesVlv100	ULINT	0,0	Number of cycles
CyclesVlv0	ULINT	0,0	Number of cycles
CyclesCoarseVlv100	ULINT	0,0	Number of cycles
CyclesCoarseVlv0	ULINT	0,0	Number of cycles
dtDateTimeUTC	DATE_AND_TIME	0,0	Save the date of last calibration
ValveOnOffTime	REAL	20.0	Time how long does the valve need to open the valve
CloseTightEnable	BOOL	TRUE	Enables CloseTight Parameter (1: enabled 0: disabled)
CloseTightThreshold	REAL	1.0	Parameter for Close Tight Threshold

Tab. 104 DrvCtrl persistent variables

16.1.8 Symbols

16.1.8.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Valve symbol	Grey: PV < 25% Blue: PV > 25%	
Alarm symbol	 : alarm is active (top priority)	See description of function block and alarm function.
Operation Mode	 : offline mode is active  : operator mode is active  : automatic mode is active	See description of operation mode.
Interlock	 : no interlock or protection lock is active  : interlock is active  : protection lock is active	See lock function description.
Permission	 : access permitted  : access denied	See lock function description.
Setpoint	Depending on the source mode, “Splnt” or “SpMan” is displayed.	Max. 5 places before and 2 places after the decimal point are displayed.
Process Value	“PV” is displayed (actual or simulated value).	Max. 5 places before and 2 places after the decimal point are displayed.
Unit	U/M “%” is displayed for the setpoint value and the process value.	Cannot be changed.
Nozzles	Nozzles for connecting logical lines (dashed nozzles) and pipes (blue nozzles).	
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 105 DrvCtrl symbol description

16.1.8.2 hDrvCtrlValue/vDrvCtrlValue

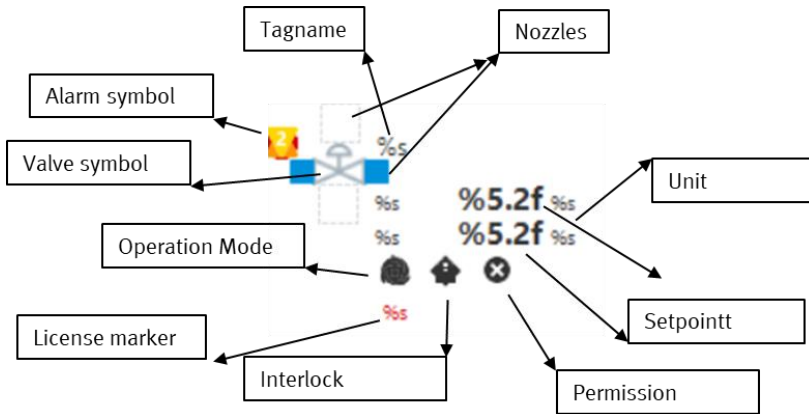


Fig. 106 Symbol hDrvCtrlValue and vDrvCtrlValue

16.1.8.3 hDrvCtrlLinear

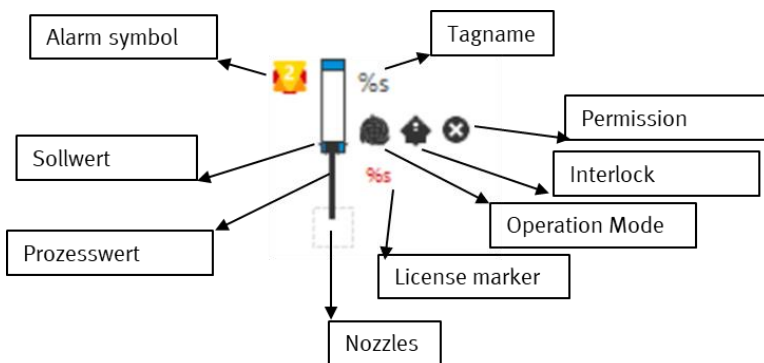


Fig. 107 Symbol hDrvCtrlLinear

16.1.9 Faceplates

16.1.9.1 Main

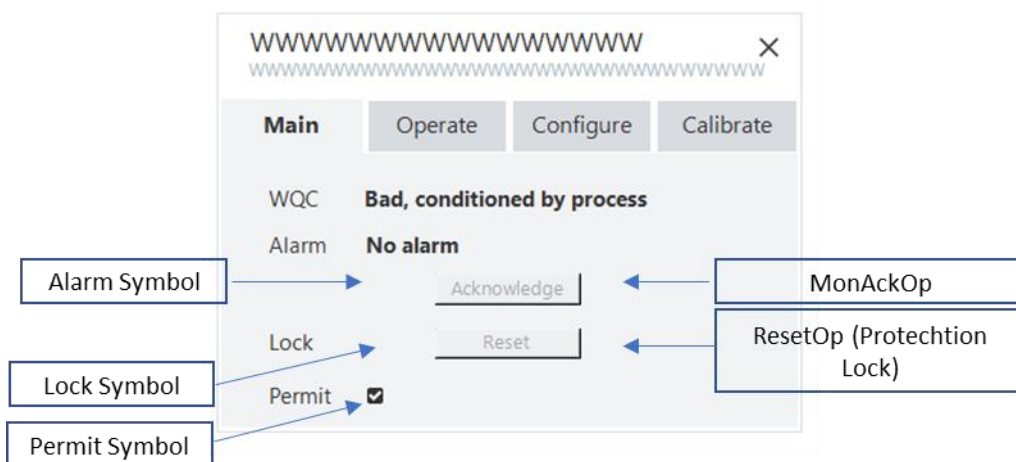


Fig. 108 DrvCtrl main faceplate

16.1.9.2 Operate

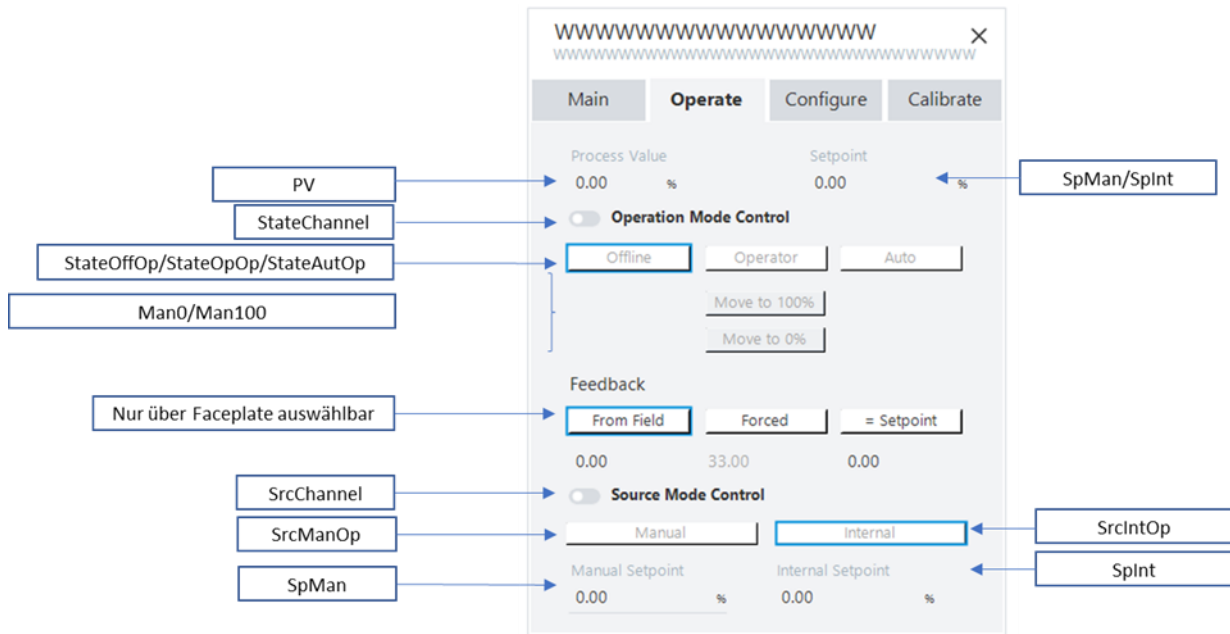


Fig. 109 DrvCtrl operate faceplate

16.1.9.3 Configure

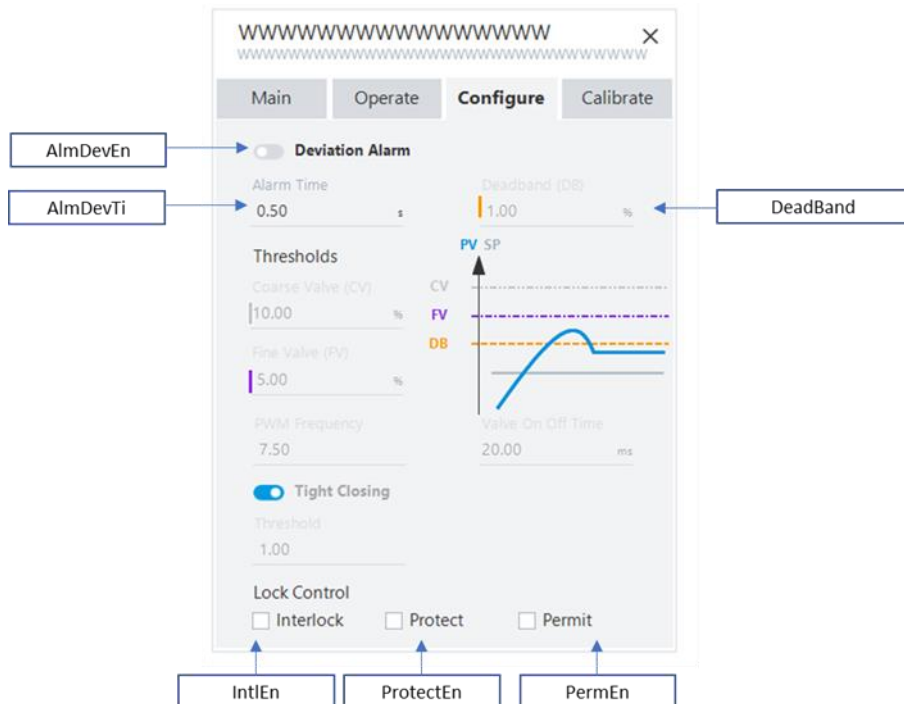


Fig. 110 DrvCtrl configure faceplate

16.1.9.4 Calibrate

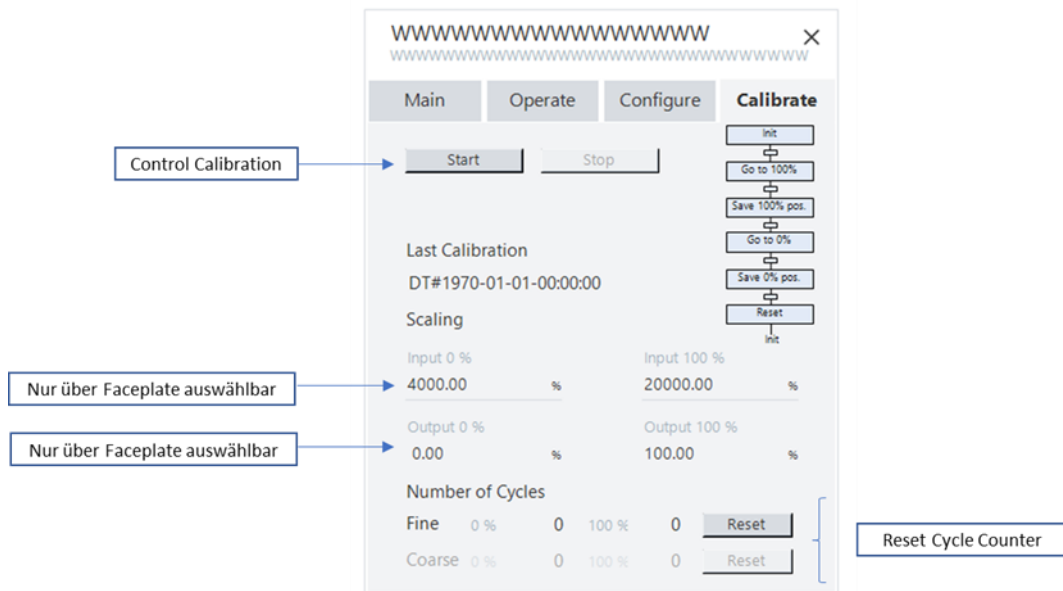


Fig. 111 DrvCtrl calibrate faceplate

17 Interlock display

17.1 LockView4 function block

17.1.1 Brief description

The LockView4 function block (FB) provides the option of implementing interlocking with conditions. The results obtained from linking all active inputs are used to set the protect, interlock and permit inputs at other FBs, and thus to disable or enable control commands.

17.1.2 Basic functions

17.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

17.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 106 Worst quality code enumeration

17.1.3 Functions

17.1.3.1 Input values "In1", "In2", "In3" and "In4"

The FB has 4 inputs which can be individually activated (In1En, In2En ...). All of the values of the active inputs are used for logic operation and generate the output value.

17.1.3.2 Input values "In1QC", "In2QC", "In3QC" and "In4QC"

A worst quality code (WQC) is assigned to each input. The WQC with the highest priority is read out at output "QCOut".

17.1.3.3 Input values “In1Inv”, “In2Inv”, “In3Inv” and “In4Inv”

If these inputs are set to TRUE, the input value (In1, In2 ...) is used as an inverted value for the logic operation.

17.1.3.4 Input values “In1Txt”, “In2Txt”, “In3Txt” and “In4Txt”

A text with 32 characters can be entered for the description of each input.

17.1.3.5 Output value “Out”

After the logic operation, the calculated value is read out to output “Out”. The worst quality code “QCOut” indicates the quality of the calculated value. If all inputs are deactivated, output “Out” reads out a value of TRUE. “Out” can be used at other FBs as an input for Permit, Protect and Interlock.

17.1.3.6 Logic

How all active inputs are linked can be specified via the “Logic” input at the FB:

Logic 1: In1 AND In2 AND In3 AND In4

If an active input is set to FALSE, “Out” is also set to FALSE and a permit, interlock or protection lock is triggered at the linked FB.

Logic 2: In1 OR In2 OR In3 OR In4

“Out” only assumes a value of FALSE after all active inputs are set to FALSE.

17.1.4 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 107 LockView4 specific worst quality code

17.1.5 Persistent variables

This FB doesn't have any persistent variables.

17.1.6 Symbols

17.1.6.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Interlock symbol	Represents the output's state.	The symbol is red when “Out” has a value of FALSE.
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 108 LockView4 symbol description

17.1.6.2 hLockView4Default

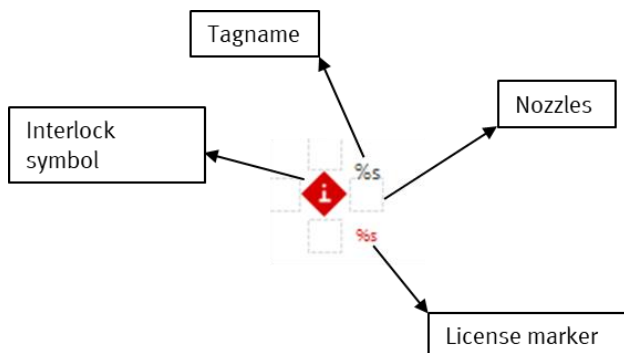


Fig. 112 Symbol hLockView4Default

17.1.7 Faceplates

17.1.7.1 Main

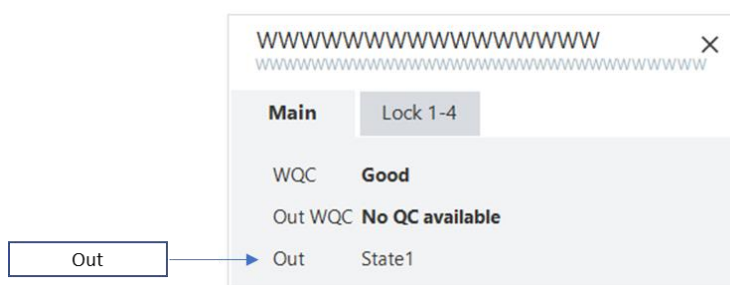


Fig. 113 LockView4 main faceplate

17.1.7.2 Lock 1-4



Fig. 114 LockView4 Lock 1-4 faceplate

17.2 LockView8 function block

17.2.1 Brief description

The LockView8 function block (FB) provides the option of implementing interlocking with conditions. The results obtained from linking all active inputs are used to set the protect, interlock and permit inputs at other FBs, and thus to disable or enable control commands.

17.2.2 Basic functions

17.2.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

17.2.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 109 Worst quality code enumeration

17.2.3 Functions

17.2.3.1 Input values "In1", "In2", "In3" "In4" ...

The FB has 8 inputs which can be individually activated (In1En, In2En ...). All of the values of the active inputs are used for logic operation and generate the output value.

17.2.3.2 Input values "In1QC", "In2QC", "In3QC", "In4QC" ...

A worst quality code (WQC) is assigned to each input. The WQC with the highest priority is read out at output "QCOut".

17.2.3.3 Input values “In1Inv”, “In2Inv”, “In3Inv”, “In4Inv” ...

If these inputs are set to TRUE, the input value (In1, In2 ...) is used as an inverted value for the logic operation.

17.2.3.4 Input values “In1Txt”, “In2Txt”, “In3Txt”, “In4Txt” ...

A text with 32 characters can be entered for the description of each input.

17.2.3.5 Output value “Out“

After the logic operation, the calculated value is read out to output “Out”. The worst quality code “QCOut” indicates the quality of the calculated value. If all inputs are deactivated, output “Out” reads out a value of TRUE. “Out” can be used at other FBs as an input for Permit, Protect and Interlock.

17.2.3.6 Logic

How all active inputs are linked can be specified via Logic:

Logic 1: In1 AND In2 AND In3 AND In4 AND In5 AND In6 AND In7 AND In8

If an active input is set to FALSE, “Out” is also set to FALSE and a permit, interlock or protection lock is triggered at the linked FB.

Logic 2: In1 OR In2 OR In3 OR In4 OR In5 OR In6 OR In7 OR In8

“Out” only assumes a value of FALSE after all active inputs are set to FALSE.

17.2.3.7 FB-specific worst quality code

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 110 LockView8 specific worst quality code

17.2.4 Persistent variables

This FB doesn't have any persistent variables.

17.2.5 Symbols

17.2.5.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Interlock symbol	Represents the output's state.	The symbol is red when “Out” has a value of FALSE.
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 111 LockView8 symbol description

17.2.5.2 hLockView8Default

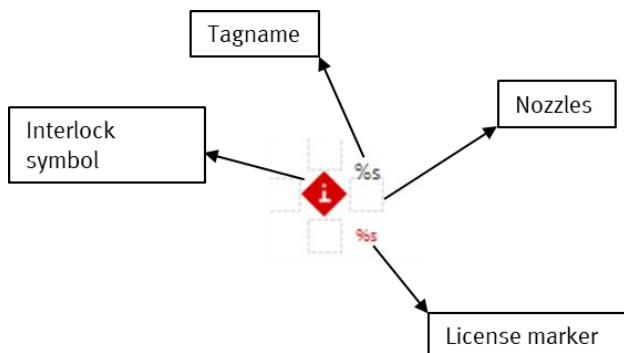


Fig. 115 Symbol hLockView8Default

17.2.6 Faceplates

17.2.6.1 Main

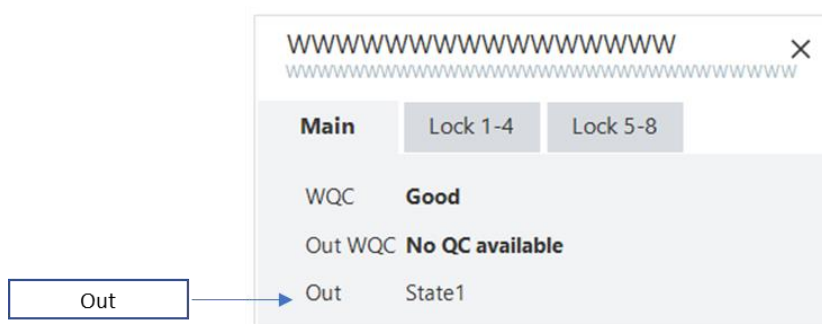


Fig. 116 LockView8 main faceplate

17.2.6.2 Lock 1-4



Fig. 117 LockView8 Lock 1-4 faceplate

17.2.6.3 Lock 5-8



Fig. 118 LockView8 Lock 5-8 faceplate

17.3 LockView16 function block

17.3.1 Brief description

The LockView16 function block (FB) provides the option of implementing interlocking with conditions. The results obtained from linking all active inputs are used to set the protect, interlock and permit inputs at other FBs, and thus to disable or enable control commands.

17.3.2 Basic functions

17.3.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

17.3.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 112 Worst quality code enumeration

17.3.3 Functions

17.3.3.1 Input values "In1", "In2", "In3" "In4" ...

The FB has 16 inputs which can be individually activated (In1En, In2En ...). All of the values of the active inputs are used for logic operation and generate the output value.

17.3.3.2 Input values "In1QC", "In2QC", "In3QC", "In4QC" ...

A worst quality code (WQC) is assigned to each input. The WQC with the highest priority is read out at output "QCOut".

17.3.3.3 Input values “In1Inv”, “In2Inv”, “In3Inv”, “In4Inv” ...

If these inputs are set to TRUE, the input value (In1, In2 ...) is used as an inverted value for the logic operation.

17.3.3.4 Input values “In1Txt”, “In2Txt”, “In3Txt”, “In4Txt” ...

A text with 32 characters can be entered for the description of each input.

17.3.3.5 Output value “Out”

After the logic operation, the calculated value is read out to output “Out”. The worst quality code “QCOut” indicates the quality of the calculated value. If all inputs are deactivated, output “Out” reads out a value of TRUE. “Out” can be used at other FBs as an input for Permit, Protect and Interlock.

17.3.3.6 Logic

How all active inputs are linked can be specified via Logic:

Logic 1: In1 AND In2 AND In3 AND In4 AND In5 AND In6 AND In7 AND In8 AND In9 AND In10 AND In11 AND In12 AND In13 AND In14 AND In15 AND In16

If an active input is set to FALSE, “Out” is also set to FALSE and a permit, interlock or protection lock is triggered at the linked FB.

Logic 2: In1 OR In2 OR In3 OR In4 OR In5 OR In6 OR In7 OR In8 OR In9 OR In10 OR In11 OR In12 OR In13 OR In14 OR In15 OR In16

“Out” only assumes a value of FALSE after all active inputs are set to FALSE.

17.3.4 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 113 LockView16 specific worst quality code

17.3.5 Persistent variables

This FB doesn't have any persistent variables.

17.3.6 Symbols

17.3.6.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Interlock symbol	Represents the output's state.	The symbol is red when “Out” has a value of FALSE.
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 114 LockView16 symbol description

17.3.6.2 hLockView16Default

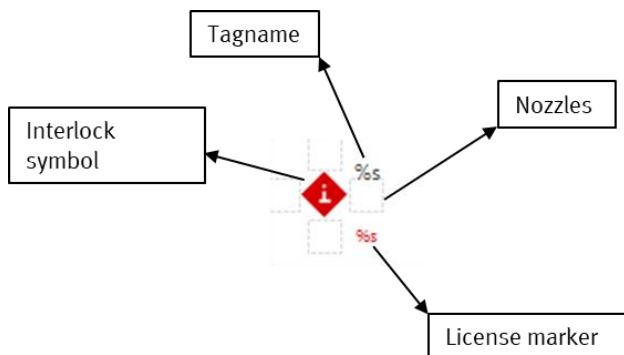


Fig. 119 Symbol hLockView16Default

17.3.7 Faceplates

17.3.7.1 Main

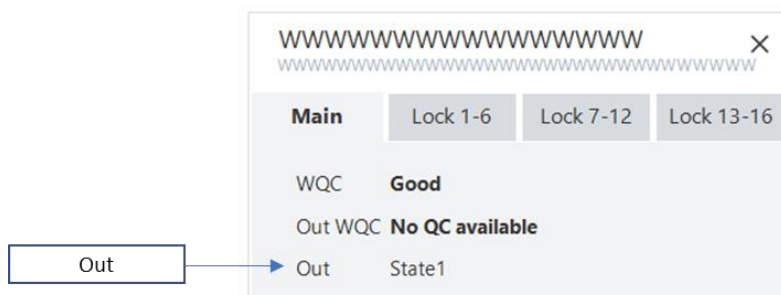


Fig. 120 LockView16 main faceplate

17.3.7.2 Lock 1-6



Fig. 121 LockView16 Lock 1-6 faceplate

17.3.7.3 Lock 7-12

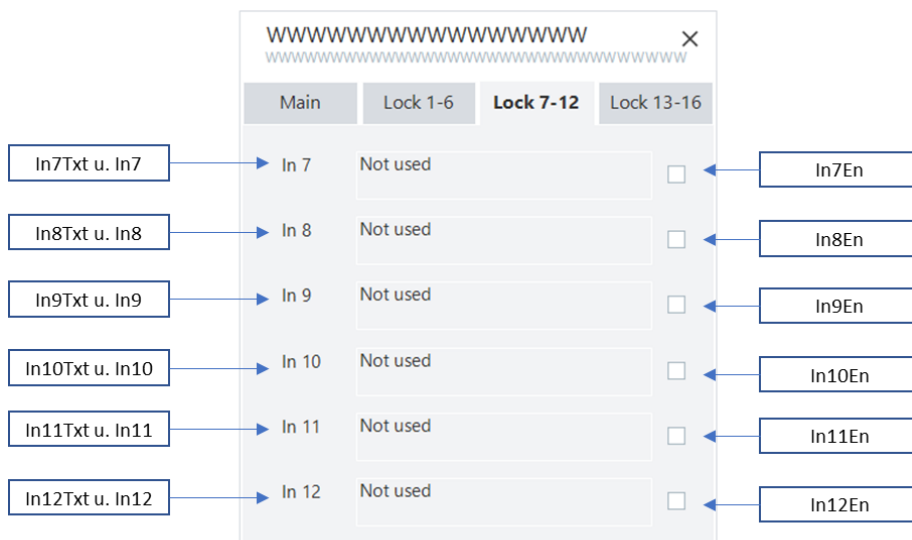


Fig. 122 LockView16 Lock 7-12 faceplate

17.3.7.4 Lock 13-16

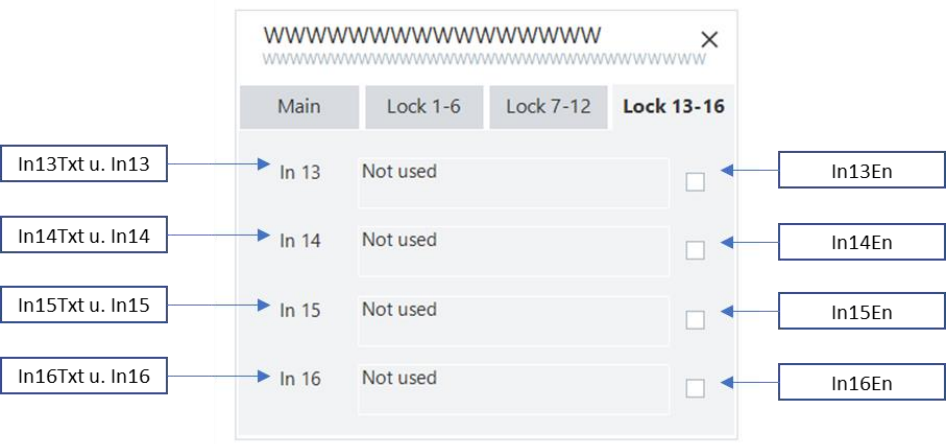
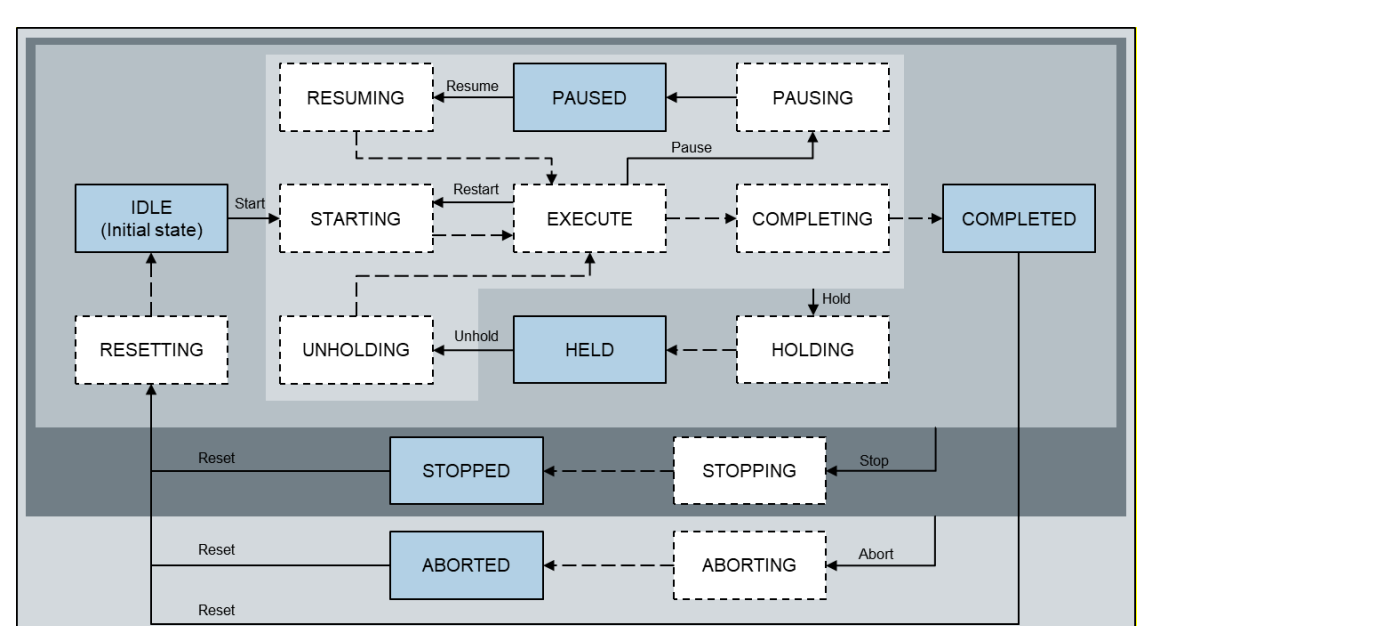


Fig. 123 LockView16 Lock 13-16 faceplate

The SimpleSelfCompletingServiceControl function block (FB) makes it possible to implement a self-completing service in accordance with a standardised state machine (see figure). The function block specifies the framework and concrete functionality can be implemented within one of the 16 states depending on the application. Transitions amongst the 16 states of the state machine are either automatic (in the case of transient states – shown with dashed lines in the figure) or are triggered by a specified control command (in the case of non-transient states – shown with solid lines in the figure).



A service itself has three states – offline, operator and automatic. The service is only ready for operation in the offline state. In contrast to this, the service is active in both the operator and automatic states, i.e. the above state machine is executed.

In the operator state, a command which triggers a state transition is specified by the operator via the CommandOp variable. For example, the automatic state is intended for use of the service within the framework of recipe processing – either an internally or an externally specified command is used in this case. The interface includes a corresponding changeover for this purpose, whereby the source mode function is used.

18.1.2 Basic functions

18.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

18.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 115 Worst quality code enumeration

18.1.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

18.1.3 Functions

18.1.3.1 Implementation and use of application- specific services

The SimpleSelfCompletingServiceControl FB provides a basic framework for a service to be offered, including the above state machine. However, the functionality which will be executed within the various states is application-specific. And thus if an application-specific service of this sort will be offered, a corresponding FB must first be set up which extends the SimpleSelfCompletingServiceControl FB:

(* declaration example *)

FUNCTION_BLOCK ServiceExample EXTENDS SimpleSelfCompletingServiceControl

(* Implementation example ST *)

SUPER^0;

The functionality of the required states can be subsequently defined for this application-specific FB. Each state is implemented as an action of the sequential function chart type (AS).

Note:

- The available functions are automatically offered for selection by means of “right click -> Add object -> Action ...”.
- States which are not required don't have to be implemented.

Attention:

In order for the basic implementation of the ServiceControl FB to be able to query the current state of the step sequences of the respective states and reset them (upon leaving the current state), access to AS flags SFCInit and SFCCurrentStep must be activated. This can be activated project-wide in the project settings (Project -> Project settings -> SFC) as follows:

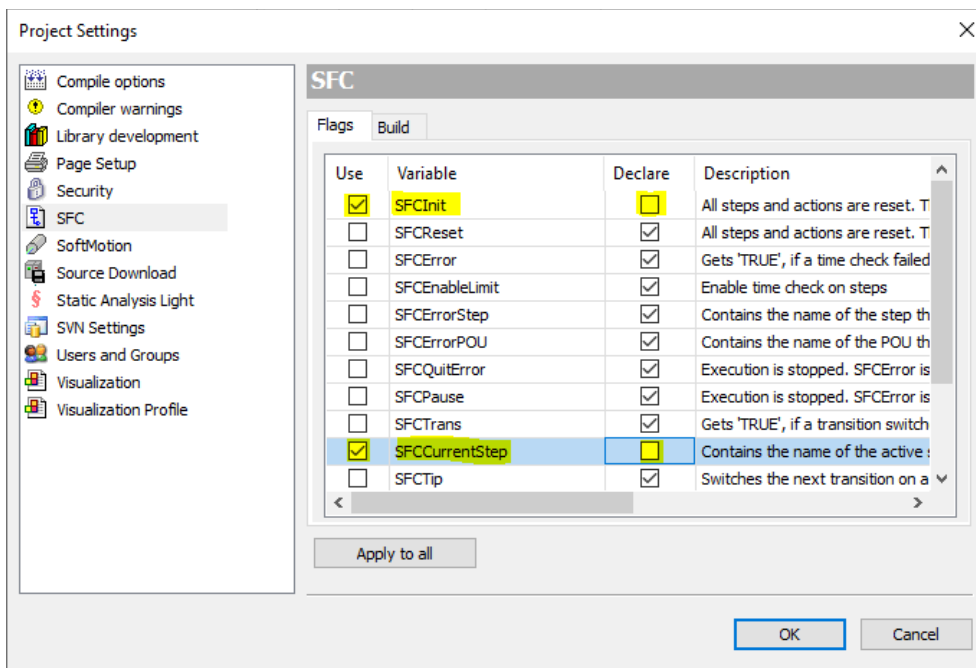


Fig. 125 Project settings

Alternatively, this setting can also be selected individually for the implemented state (in the properties of the state action) (right click -> Properties... -> SFC settings):

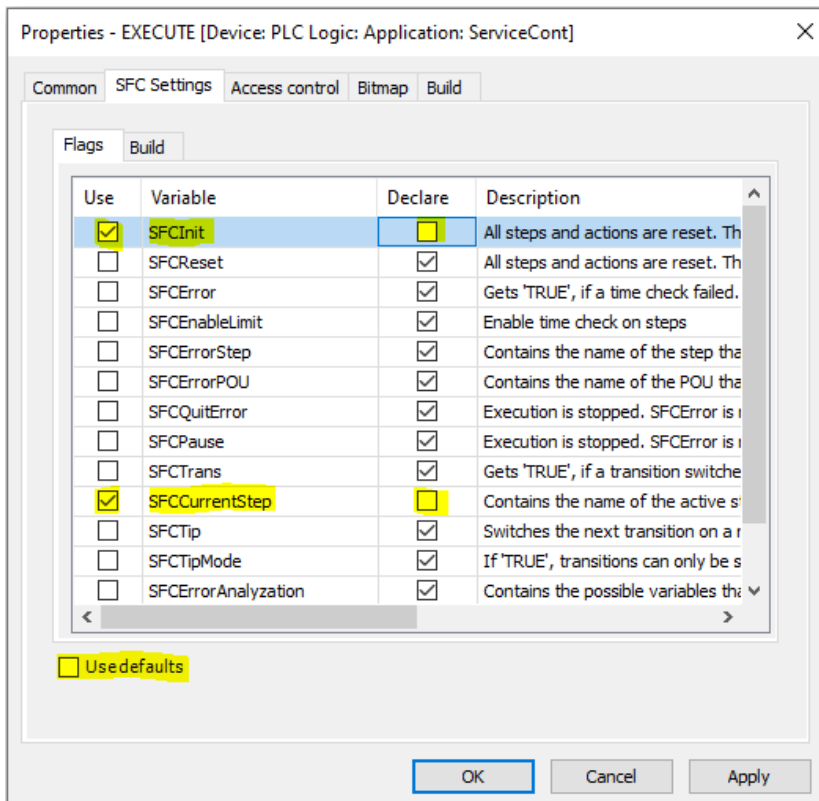


Fig. 126 Individual settings

The resulting FB can then be instantiated and used as usual:

```
(* Declaration example *)
VAR
    Instance_ServiceExample: ServiceExample;
END_VAR
(* Implementation exmaple ST *)
Instance_ServiceExample();
```

18.1.3.2 Transient and not transient states

Transient states (shown in the figure above with dashed lines) automatically result in a transition to the following state after being executed once, i.e. as soon as jumping back to the initial step of the assigned AS implementation occurs. In contrast to this, non-transient states (shown in the figure above with solid lines) are not left until a corresponding command is issued, For example, by the operator. And thus a loop can also be implemented in a non-transient state, which runs through the initial state several times.

18.1.3.3 Preventing state transitions

Fundamentally, state transitions are active amongst the 16 states in accordance with the figure above. However, in some cases it may make good sense to temporarily block certain transitions (e.g. in order to

prevent simultaneous activation of two services which access the same actuators). The corresponding Prevent...Command inputs are available to this end.

18.1.3.4 Display of the current sub step in the visualisation

Fundamentally, the currently active state is displayed in a service's faceplate (e.g. Execute). However, due to the fact that complex algorithms with several sub-steps may be executed in any given state, it may make good sense to provide the operator with more precise information concerning the current processing step via the faceplate. Any desired character string with up to 24 characters can be entered via the ActualStep input to this end, which is then also displayed in the faceplate. If the texts used for this purpose are managed in a text list and are also available to a higher level system (For example, a recipe manager), the ID of the currently displayed text can also be entered via the PosTextID input. Due to the fact that this variable can be read via OPC UA, this mechanism makes it possible to display the current processing step in a higher level system of this sort as well.

18.1.3.5 Option for querying the operator

The InteractQuestionID input is used to implement queries to the operator in the form of a simple dialogue with predefined questions and possible answers. The ID of the respective question can be selected from a text list. The operator is then able to respond to the question by selecting one of several predefined answers. The answer can then be evaluated via internal variable InteractAnswerID, For example, within the framework of the implementation of one of the service's states.

18.1.3.6 Preparing and enabling utilised sensors/actuators

In nearly all cases, a service acts upon one or more sensors and/or actuators (For example, instances of the AnaView or AnaDrv FB). And thus it must be assured that the sensors and actuators are in a correspondingly serviceable state, before the service switches from the offline to the operator or automatic state. The function block defines the regainActuatorControls message to this end, which is invoked within the framework of a corresponding state transition. If this method is overwritten, special algorithms can be implemented – For example, in order to switch all utilised sensors/actuators to the operator or automatic state as well.

Analogously, the releaseActuatorControls method is also available in order to enable utilised sensors/actuators if necessary, as soon as the service is deactivated and transitioned to the offline state.

Note:

- Both methods can be selected automatically by means of “right click -> Add object -> Method ...”.

18.1.4 Extended functions

18.1.4.1 Using procedures

The SimpleSelfCompletingServiceControl function block extends the ServiceControl function block and thus also includes the option of defining and selecting various procedures. Please refer to the documentation for the ServiceControl and ServiceProcedure function blocks for in-depth explanations concerning this subject matter.

Internally, the function block already provides a (default) procedure with ID 1, which is continuously activated as long as no other procedure has been defined.

18.1.5 Source mode

18.1.5.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (external). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (VInt) the “SrcIntAct” output is set to TRUE. When the manual source is active (VExt), the “SrcExtAct” output bit is set to TRUE.

The internal source is active as a default setting.

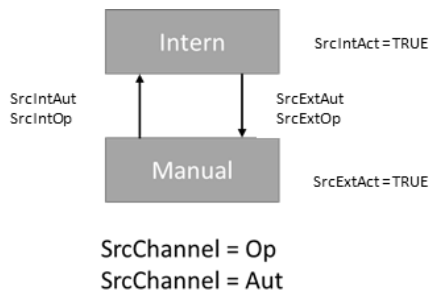


Fig. 127 Statemachine source mode

18.1.5.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcExtAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAut” and “SrcExtAut” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn't necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they're acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcExtAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

18.1.5.3 Module-external access (OPC UA)

The “SrcIntOp” and “SrcExtOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit's faceplate).
- Set “SrcIntOp” (internal source) or “SrcExtOp” (external source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcExtOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn't necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they're acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntOp” and “SrcExtOp”) are set simultaneously, “SrcIntOp” has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “SrcIntOp”, “SrcExtOp”, “SrcIntAut” or “SrcExtAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it's important to ensure that these inputs are not permanently assigned to the FB.

18.1.6 Operation mode

18.1.6.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode.

The “StateChannel” input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the “StateOffAct” output is set to TRUE. When the operator mode is active “StateOpAct” is set to TRUE, and when the automatic mode is active “StateAutAct” is set to TRUE.

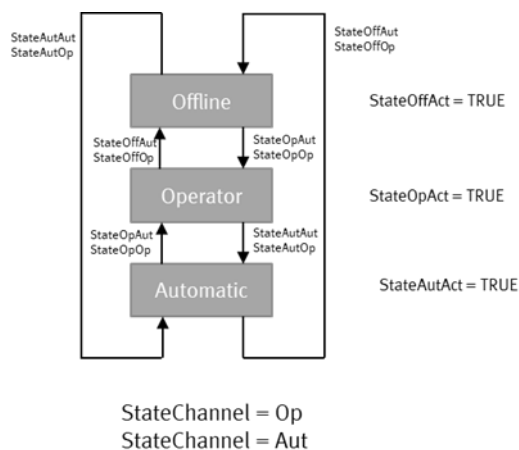


Fig. 128 Statemachine operation mode

18.1.6.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- “StateChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “StateOffAut” input (switch to offline mode), the “StateOpAut” input (switch to operator mode) or the “StateAutAut” input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the source is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

18.1.6.3 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.

- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

18.2 SimpleContinuousServiceControl

18.2.1 Brief description

The SimpleContinuousServiceControl function block (FB) makes it possible to implement a continuous service in accordance with a standardised state machine (see figure). The function block specifies the framework and concrete functionality can be implemented within any one of the 16 states depending on the application. Transitions amongst the 16 states of the state machine are either automatic (in the case of transient states – shown with dashed lines in the figure) or are triggered by a specified control command (in the case of non-transient states – shown with solid lines in the figure).

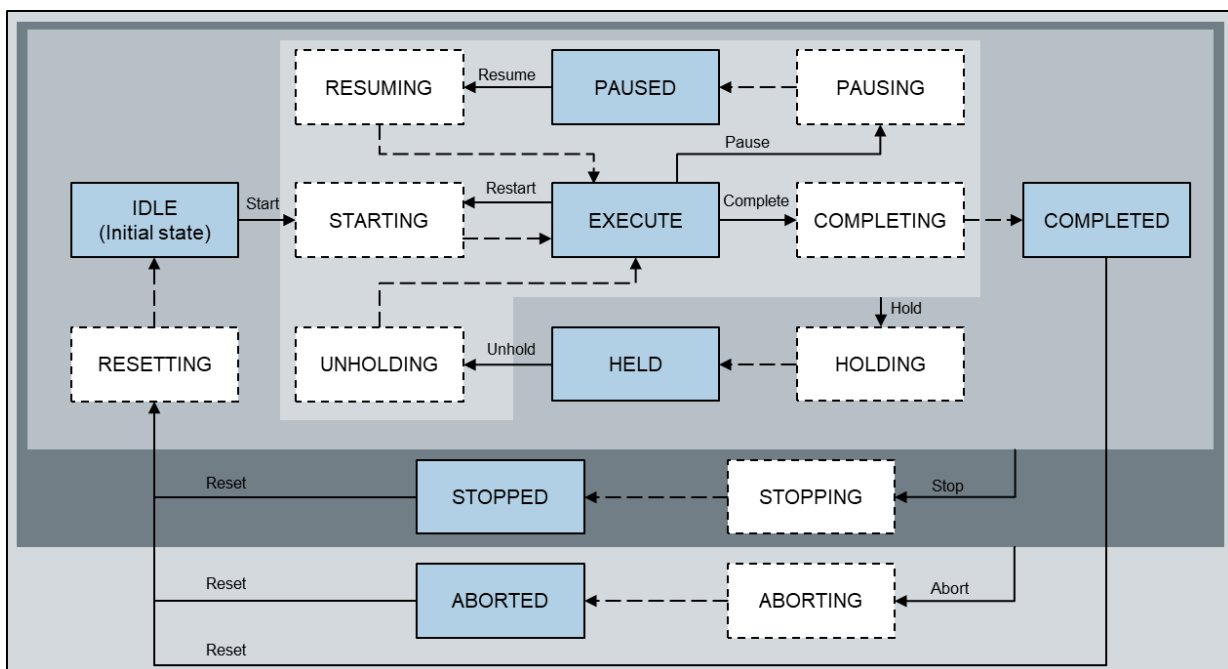


Fig. 129 Statemachine SimpleCotinuousServiceControl

A service itself has three states – offline, operator and automatic. The service is only ready for operation in the offline state. In contrast to this, the service is active in both the operator and automatic states, i.e. the above state machine is executed.

In the operator state, a command which triggers a state transition is specified by the operator via the CommandOp variable. For example, the automatic state is intended for use of the service within the framework of recipe processing – either an internally or an externally specified command is used in this case. The interface includes a corresponding changeover for this purpose, whereby the source mode function is used.

18.2.2 Basic functions

18.2.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

18.2.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 116 Worst quality code enumeration

18.2.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

18.2.3 Functions

18.2.3.1 Implementation and use of application-specific services

The SimpleContinuousServiceControl FB provides a basic framework for a service to be offered, including the above state machine. However, the functionality which will be executed within the various states is application-specific. And thus if an application-specific service of this sort will be offered, a corresponding FB must first be set up which extends the SimpleContinuousServiceControlFB:

(* Declaration example *)

```
FUNCTION_BLOCK ServiceExample EXTENDS SimpleContinuousServiceControl
```

(* Implementation example ST *)

```
SUPER^0;
```

The functionality of the required states can be subsequently defined for this application-specific FB. Each state is implemented as an action of the sequential function chart type (AS).

Note:

- The available functions are automatically offered for selection by means of “right click -> Add object -> Action ...”.
- States which are not required don’t have to be implemented.

Attention:

In order for the basic implementation of the ServiceControl FB to be able to query the current state of the step sequences of the respective states and reset them (upon leaving the current state), access to AS flags SFCInit and SFCCurrentStep must be activated. This can be activated project-wide in the project settings (Project -> Project settings -> SFC) as follows:

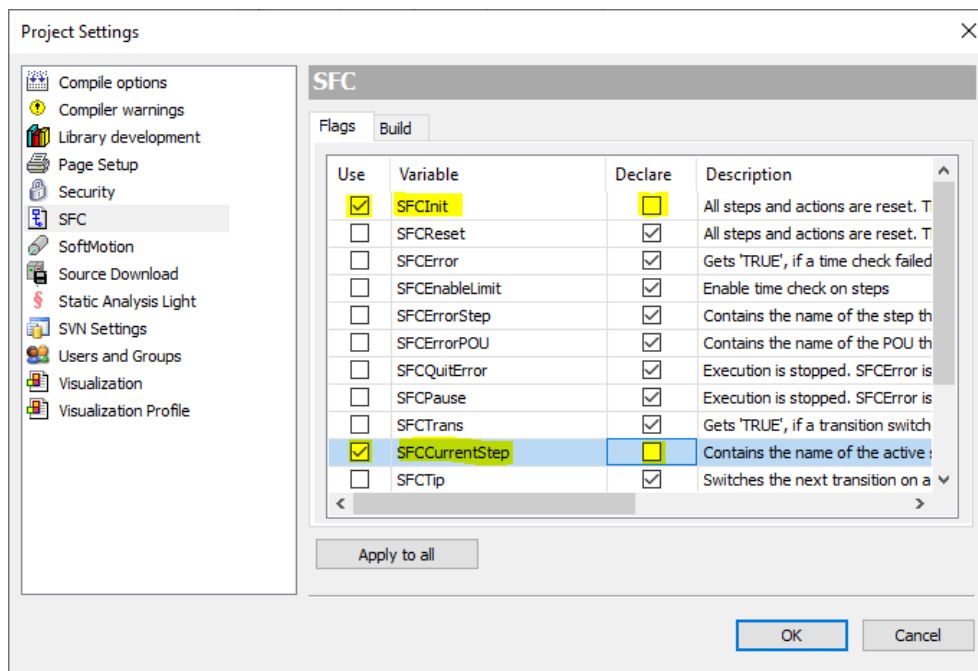


Fig. 130 Project settings

Alternatively, this setting can also be selected individually for the implemented state (in the properties of the state action) (right click -> Properties... -> SFC settings):

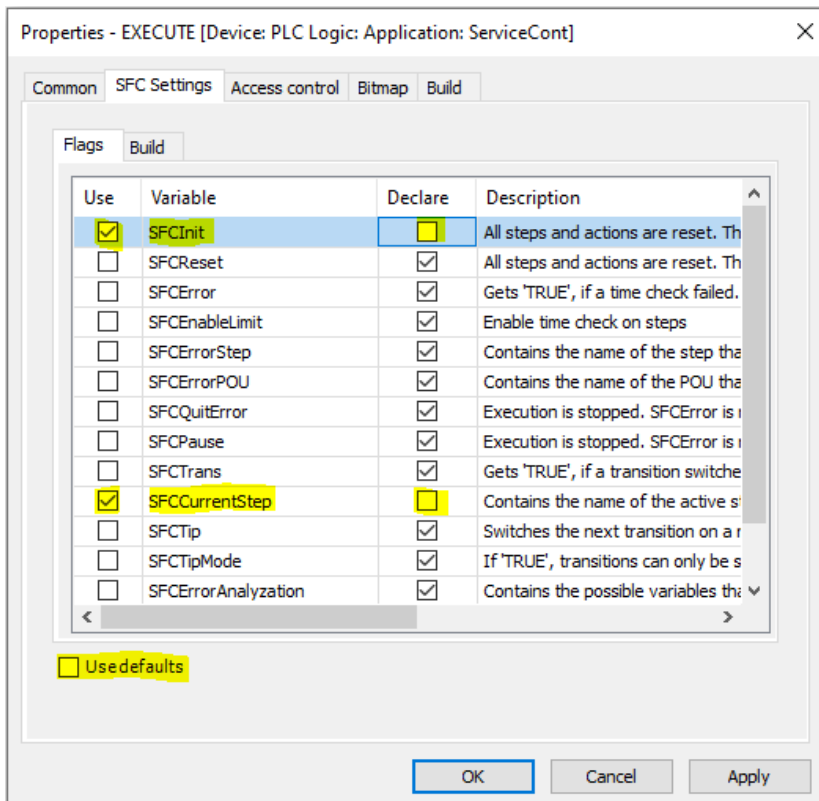


Fig. 131 Individual settings

The resulting FB can then be instantiated and used as usual:

(* Declaration example *)

VAR

Instance_ServiceExample: ServiceExample;

END_VAR

(* Implementation example ST *)

Instance_ServiceExample();

18.2.3.2 Transient and not transient states

Transient states (shown in the figure above with dashed lines) automatically result in a transition to the following state after being executed once, i.e. as soon as jumping back to the initial step of the assigned AS implementation occurs. In contrast to this, non-transient states (shown in the figure above with solid lines) are not left until a corresponding command is issued, For example, by the operator. And thus a loop can also be implemented in a non-transient state, which runs through the initial state several times.

18.2.3.3 Preventing state transitions

Fundamentally, state transitions are active amongst the 16 states in accordance with the figure above. However, in some cases it may make good sense to temporarily block certain transitions (e.g. in order to

prevent simultaneous activation of two services which access the same actuators). The corresponding Prevent...Command inputs are available to this end.

18.2.3.4 Display of the current sub step in the visualisation

Fundamentally, the currently active state is displayed in a service's faceplate (e.g. Execute). However, due to the fact that complex algorithms with several sub-steps may be executed in any given state, it may make good sense to provide the operator with more precise information concerning the current processing step via the faceplate. Any desired character string with up to 24 characters can be entered via the ActualStep input to this end, which is then also displayed in the faceplate. If the texts used for this purpose are managed in a text list and are also available to a higher level system (For example, a recipe manager), the ID of the currently displayed text can also be entered via the PosTextID input. Due to the fact that this variable can be read via OPC UA, this mechanism makes it possible to display the current processing step in a higher level system of this sort as well.

18.2.3.5 Option for querying the operator

The InteractQuestionID input is used to implement queries to the operator in the form of a simple dialogue with predefined questions and possible answers. The ID of the respective question can be selected from a text list. The operator is then able to respond to the question by selecting one of several predefined answers. The answer can then be evaluated via internal variable InteractAnswerID, For example, within the framework of the implementation of one of the service's states.

18.2.3.6 Preparing and enabling utilised sensors/actuators

In nearly all cases, a service acts upon one or more sensors and/or actuators (For example, instances of the AnaView or AnaDrv FB). And thus it must be assured that the sensors and actuators are in a correspondingly serviceable state, before the service switches from the offline to the operator or automatic state. The function block defines the regainActuatorControls message to this end, which is invoked within the framework of a corresponding state transition. If this method is overwritten, special algorithms can be implemented – For example, in order to switch all utilised sensors/actuators to the operator or automatic state as well. Analogously, the releaseActuatorControls method is also available in order to enable utilised sensors/actuators if necessary, as soon as the service is deactivated and transitioned to the offline state.

Note:

- Both methods can be selected automatically by means of “right click -> Add object -> Method ...”.

18.2.4 Extended functions

18.2.4.1 Using procedures

The SimpleSelfCompletingServiceControl function block extends the ServiceControl function block and thus also includes the option of defining and selecting various procedures. Please refer to the documentation for the ServiceControl and ServiceProcedure function blocks for in-depth explanations concerning this subject matter.

Internally, the function block already provides a (default) procedure with ID 1, which is continuously activated as long as no other procedure has been defined.

18.2.5 Source mode

18.2.5.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (external). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (VInt) the “SrcIntAct” output is set to TRUE. When the manual source is active (VExt), the “SrcExtAct” output bit is set to TRUE.

The internal source is active as a default setting.

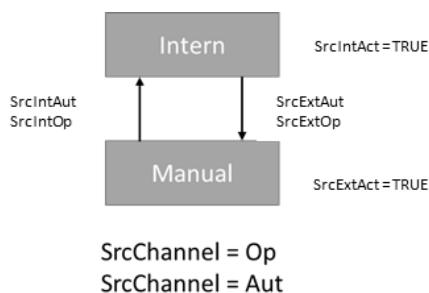


Fig. 132 Statemachine source mode

18.2.5.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcExtAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAut” and “SrcExtAut” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn't necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they're acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcExtAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

18.2.5.3 Module-external access (OPC UA)

The “SrcIntOp” and “SrcExtOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit's faceplate).
- Set “SrcIntOp” (internal source) or “SrcExtOp” (external source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcExtOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn't necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they're acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntOp” and “SrcExtOp”) are set simultaneously, “SrcIntOp” has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “SrcIntOp”, “SrcExtOp”, “SrcIntAut” or “SrcExtAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it's important to ensure that these inputs are not permanently assigned to the FB.

18.2.6 Operation mode

18.2.6.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode.

The “StateChannel” input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the “StateOffAct” output is set to TRUE. When the operator mode is active “StateOpAct” is set to TRUE, and when the automatic mode is active “StateAutAct” is set to TRUE.

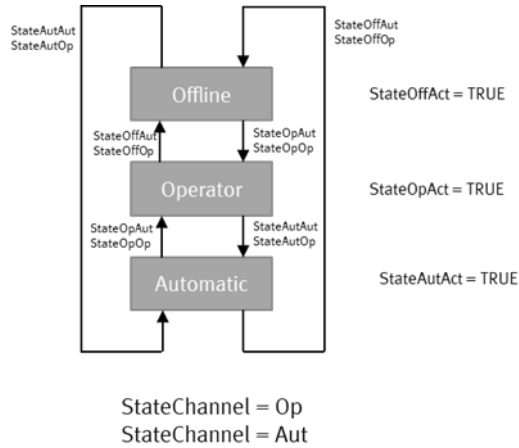


Fig. 133 Statemachine operation mode

18.2.6.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- “StateChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “StateOffAut” input (switch to offline mode), the “StateOpAut” input (switch to operator mode) or the “StateAutAut” input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the source is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

18.2.6.3 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.
- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

18.3 ServiceControl

18.3.1 Brief description

The ServiceControl function block (FB) makes it possible to implement a complex service in accordance with a standardised state machine (see figure). The function block specifies the framework and concrete functionality can be implemented within any one of the 16 states depending on the application. Transitions amongst the 16 states of the state machine are either automatic (in the case of transient states – shown with dashed lines in the figure) or are triggered by a specified control command (in the case of non-transient states – shown with solid lines in the figure).

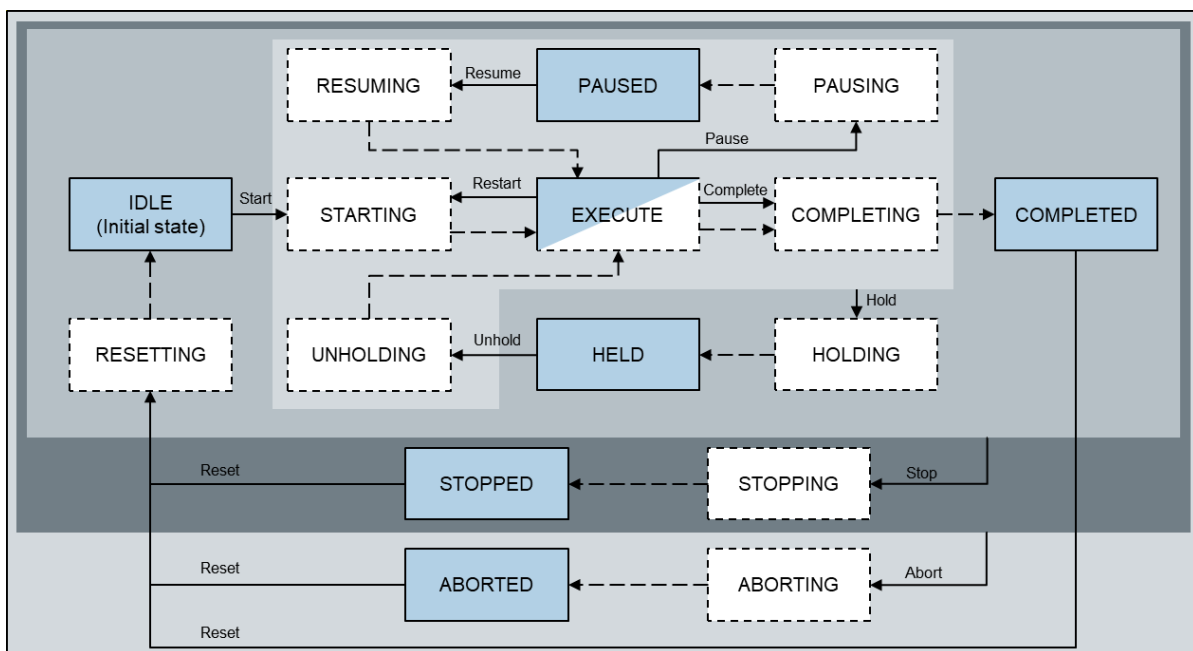


Fig. 134 State machine ServiceControl

A service itself has three states – offline, operator and automatic. The service is only ready for operation in the offline state. In contrast to this, the service is active in both the operator and automatic states, i.e. the above state machine is executed.

In the operator state, a command which triggers a state transition is specified by the operator via the CommandOp variable. For example, the automatic state is intended for use of the service within the framework of recipe processing – either an internally or an externally specified command is used in this case. The interface includes a corresponding changeover for this purpose, whereby the source mode function is used.

A service may include several procedures or modes of operation, which represent various versions of the service. Analogous to triggering a command, three variables are available for selecting a procedure to be activated, namely ProcedureOp (in the Operator state) and ProcedureInt/Ext (in the Automatic state).

It must be assured that procedures are only activated in the Starting state. If the value of the active variable for selecting the procedure changes, For example, to the Execute state, the procedure is not activated until the service is restarted (and corresponding execution of the Starting state).

18.3.2 Basic functions

18.3.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

18.3.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 117 Worst quality code enumeration

18.3.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

18.3.3 Functions

18.3.3.1 Implementation and use of application-specific services

The ServiceControl FB provides a basic framework for a service to be offered, including the above state machine. However, the functionality which will be executed within the various states is application-specific. And thus if an application-specific service of this sort will be offered, a corresponding FB must first be set up which extends the ServiceControl FB:

```
(* Declaration example *)  
FUNCTION_BLOCK ServiceExample EXTENDS ServiceControl  
(* Implementation example ST *)  
SUPER^();
```

The functionality of the required states can be subsequently defined for this application-specific FB. Each state is implemented as an action of the sequential function chart type (AS).

Note:

- The available functions are automatically offered for selection by means of “right click -> Add object -> Action ...”.
- States which are not required don’t have to be implemented.

Attention:

In order for the basic implementation of the ServiceControl FB to be able to query the current state of the step sequences of the respective states and reset them (upon leaving the current state), access to AS flags SFCInit and SFCCurrentStep must be activated. This can be activated project-wide in the project settings (Project -> Project settings -> SFC) as follows:

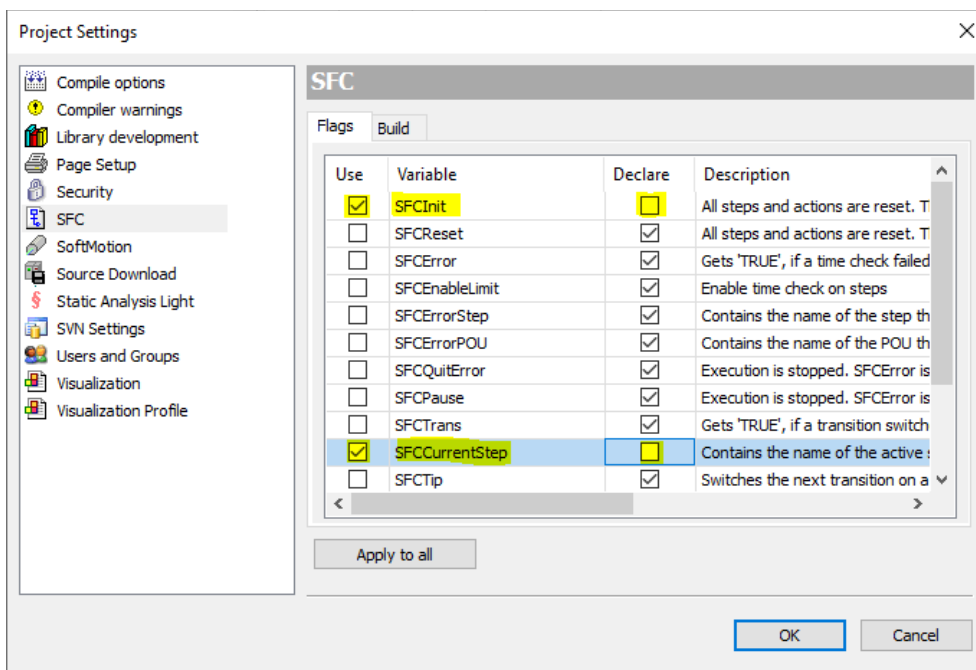


Fig. 135 Project settings

Alternatively, this setting can also be selected individually for the implemented state (in the properties of the state action) (right click -> Properties... -> SFC settings):

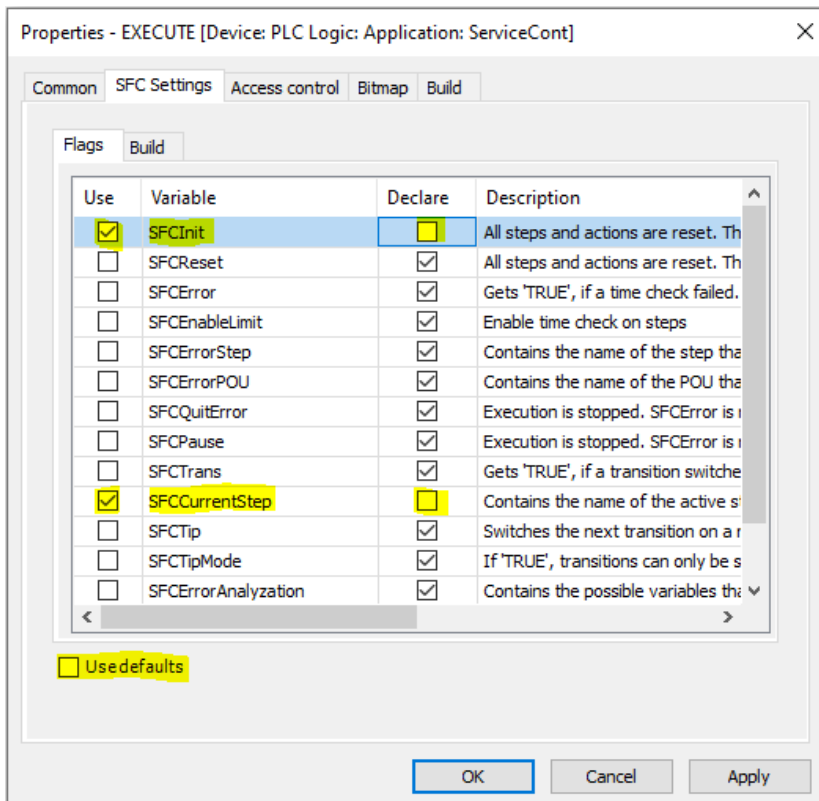


Fig. 136 Individual settings

The resulting FB can then be instantiated and used as usual:

```
(* Declaration example *)
VAR
    Instance_ServiceExample : ServiceExample;
END_VAR
(* Implementation example ST *)
Instance_ServiceExample();
```

18.3.3.2 Specification and selection of procedures

Each service can define any desired number of procedures, and in doing so instantiate the ServiceProcedure FB (for further information, please refer to the documentation for this FB). However, it's absolutely essential to make a default procedure available with ID 1. It's also mandatory to overwrite the isProcedureIDValid and isProcedureContinuous methods. IsProcedureIDValid informs the service as to whether or not any given requested procedure exists. If this is not the case, the service switches automatically to the default procedure (with ID 1). In contrast, the isProcedureContinuous indicates whether or not a procedure is self-completing or continuous and controls, amongst other things, enabling and triggering of the transition between the Execute and Completing states.

Note:

Both methods can be selected automatically by means of "right click -> Add object -> Method ...".

18.3.3.3 Transient and not transient states

Transient states (shown in the figure above with dashed lines) automatically result in a transition to the following state after being executed once, i.e. as soon as jumping back to the initial step of the assigned AS implementation occurs. In contrast to this, non-transient states (shown in the figure above with solid lines) are not left until a corresponding command is issued, For example, by the operator. And thus a loop can also be implemented in a non-transient state, which runs through the initial state several times.

The Execute state represents a special case in this regard which, depending on the type of currently active procedure, can be either continuous or self-completing.

18.3.3.4 Preventing state transitions

Fundamentally, state transitions are active amongst the 16 states in accordance with the figure above. However, in some cases it may make good sense to temporarily block certain transitions (e.g. in order to prevent simultaneous activation of two services which access the same actuators). The corresponding Prevent...Command inputs are available to this end.

18.3.3.5 Display of the current sub step in the visualisation

Fundamentally, the currently active state is displayed in a service's faceplate (e.g. Execute). However, due to the fact that complex algorithms with several sub-steps may be executed in any given state, it may make good sense to provide the operator with more precise information concerning the current processing step via the faceplate. Any desired character string with up to 24 characters can be entered via the ActualStep input to this end, which is then also displayed in the faceplate. If the texts used for this purpose are managed in a text list and are also available to a higher level system (For example, a recipe manager), the ID of the currently displayed text can also be entered via the PosTextID input. Due to the fact that this variable can be read via OPC UA, this mechanism makes it possible to display the current processing step in a higher level system of this sort as well.

18.3.3.6 Option for querying the operator

The InteractQuestionID input is used to implement queries to the operator in the form of a simple dialogue with predefined questions and possible answers. The ID of the respective question can be selected from a text list. The operator is then able to respond to the question by selecting one of several predefined answers. The answer can then be evaluated via internal variable InteractAnswerID, For example, within the framework of the implementation of one of the service's states.

18.3.3.7 Preparing and enabling utilised sensors/actuators

In nearly all cases, a service acts upon one or more sensors and/or actuators (For example, instances of the AnaView or AnaDrv FB). And thus it must be assured that the sensors and actuators are in a correspondingly serviceable state, before the service switches from the offline to the operator or automatic state. The function block defines the regainActuatorControls message to this end, which is invoked within the framework of a corresponding state transition. If this method is overwritten, special algorithms can be implemented – For example, in order to switch all utilised sensors/actuators to the operator or automatic state as well.

Analogously, the releaseActuatorControls method is also available in order to enable utilised sensors/actuators if necessary, as soon as the service is deactivated and transitioned to the offline state.

Note:

Both methods can be selected automatically by means of “right click -> Add object -> Method ...”.

18.3.4 Source mode

18.3.4.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (external). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (VInt) the “SrcIntAct” output is set to TRUE. When the manual source is active (VExt), the “SrcExtAct” output bit is set to TRUE.

The internal source is active as a default setting.

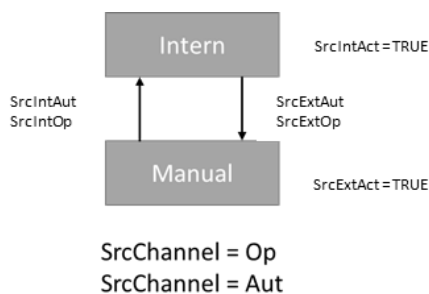


Fig. 137 Statemachine source mode

18.3.4.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutoOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcExtAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAct” and “SrcExtAct” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcExtAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

18.3.4.3 Module-external access (OPC UA)

The “SrcIntOp” and “SrcExtOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “SrcIntOp” (internal source) or “SrcExtOp” (external source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcExtOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntOp” and “SrcExtOp”) are set simultaneously, “SrcIntOp” has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “SrcIntOp”, “SrcExtOp”, “SrcIntAut” or “SrcExtAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

18.3.5 Operation mode

18.3.5.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode.

The “StateChannel” input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the “StateOffAct” output is set to TRUE. When the operator mode is active “StateOpAct” is set to TRUE, and when the automatic mode is active “StateAutAct” is set to TRUE.

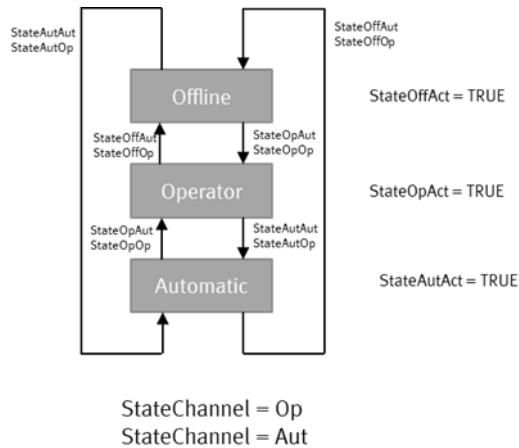


Fig. 138 Statemachine operation mode

18.3.5.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- “StateChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “StateOffAut” input (switch to offline mode), the “StateOpAut” input (switch to operator mode) or the “StateAutAut” input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the source is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

18.3.5.3 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.
- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.

- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn't necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they're acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it's not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it's important to ensure that these inputs are not permanently assigned to the FB.

18.4 Symbols

18.4.1 Description

Element	Description	Rule
Tagname	Name of the U/M – indicated in the visualisation.	Max. 16 characters.
Operation Mode	⏻ : offline mode is active 👤 : operator mode is active 🔄 : automatic mode is active	See description of operation mode.
Current State	Reads out the current state of the state machine.	
Actual Step	Text which can be implemented in the individual states is shown here.	Up to 24 characters are permissible.
License marker	If no valid licence is found for the PA Toolkit and the 2-hour limit for demo operation has expired, the FB is set to a safe state and “No licence found!” appears in the symbol.	Invisible until the licence has expired.

Tab. 118 Service symbol description

18.4.1.1 hServiceDefault

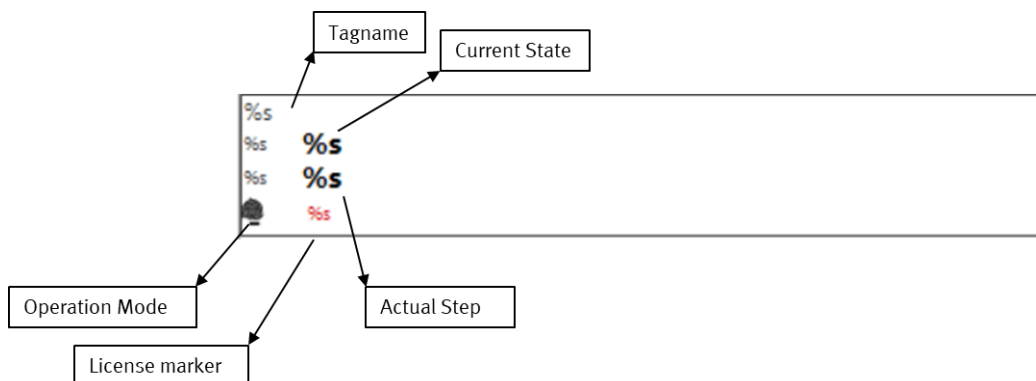


Fig. 139 Symbol hServiceDefault

18.5 Faceplates

18.5.1 Main

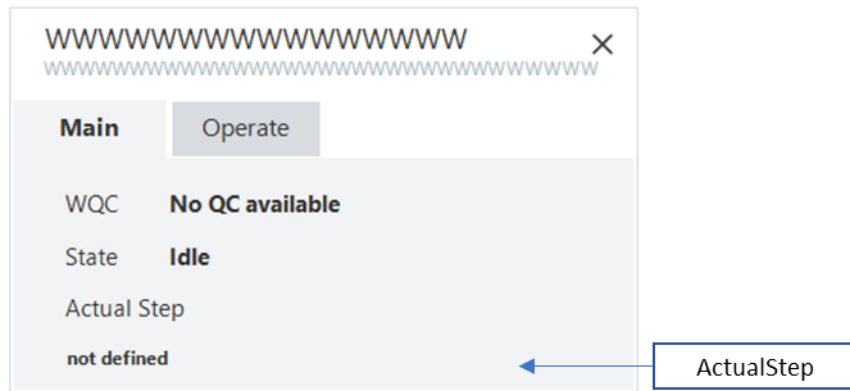


Fig. 140 Service main faceplate

18.5.2 Operate

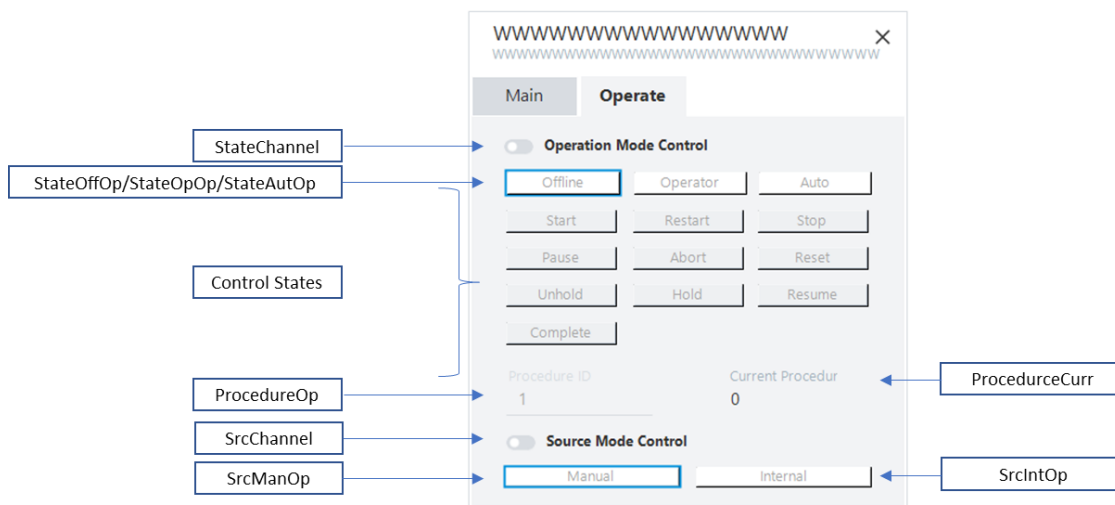


Fig. 141 Service operate faceplate

19 ServiceProcedures

19.1 ServiceProcedure

19.1.1 Brief description

A procedure or operating mode represents a specific version of a service. For example, a stirring service can include both the *continuous* and the *time-limited* procedures. The ServiceProcedure function block (FB) can be used in order to define a procedure of this sort.

19.1.2 Basic functions

19.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

19.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 119 Worst quality code enumeration

19.1.3 Functions

19.1.3.1 Characterisation of procedures

A procedure is unequivocally defined by an ID (within the scope of the service for which the procedure is defined) beschrieben. This ID is used, for example, when the procedure for a service is requested (via one of the inputs of the ServiceControl FB, namely ProcedureOp/Int/Ext). The ProcedureID input is used to specify this ID.

Note:

It's absolutely essential to assure that a procedure with ID 1 is defined for each service, which serves as a default procedure.

In addition to this, it's also possible to specify whether a procedure is continuous or self-completing via the `IsSelfCompleting` input. In the case of self-completing procedures, transition between the `Execute` and `Completing` is automatic (see documentation for the `ServiceControl` FB), whereas in the case of continuous procedures this transition can only be triggered by a corresponding command (For example, issued by the operator).

19.1.3.2 Service-procedure assignment

Fundamentally, there's an unequivocal service-procedure assignment. In order to make this assignment plainly apparent, the procedure must always be declared and initialised in the declaration section of the corresponding service FB. When invoked, the procedure FB must also be informed of the associated service via the `Service` input. The following source excerpt shows an example of declaration, initialisation and invocation of two procedures.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
  Continuously: ServiceProcedure := (TagName := 'Continuously', TagDescription := '', ProcedureID := 1,
  IsSelfCompleting := FALSE);
  Timelimited : ServiceProcedure := (TagName := 'Time-limited', TagDescription := '', ProcedureID := 2,
  IsSelfCompleting := TRUE);
END_VAR

(* Implementation example ST *)
SUPER^();
Continuously(Service := THIS^);
Timelimited(Service := THIS^);
```

19.1.3.3 Querying of the active procedure upon implementation of a service

A procedure function block indicates whether or not the procedure which it represents is active or not via the `IsActive` output. This output can be evaluated within the scope of implementation of a service, in order to effect the actual implementation of the logic for a procedure, as shown by the following example:

```
(* Implementation example execute state of stir service *)
```

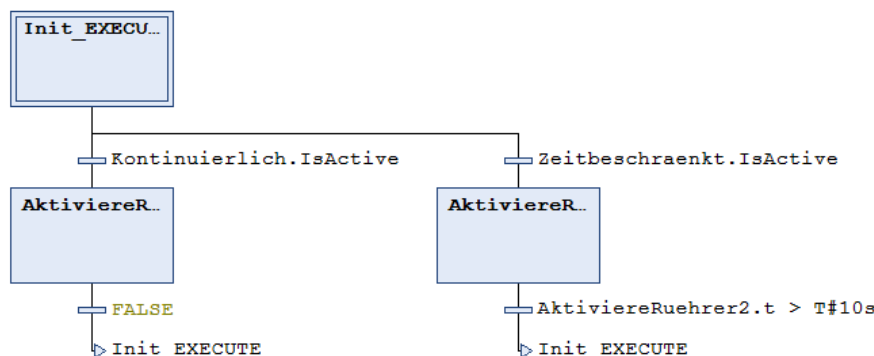


Fig. 142 Example state execute

20 Analogue service parameters

20.1 AnaConfigParam

20.1.1 Brief description

The AnaConfigParam function block (FB) can be used to define an analogue configuration parameter (floating-point number) for a service. The parameter's current value is available at the VOut output. As opposed to a simple value specification, For example, when using a VAR_INPUT of the REAL type, this provides for the following extensions:

- Analogous to a service function block, this function block also has its own state (Offline/Operator/Automatic) and permits value specifications from various sources (Operator/Internal/External).
- VOut is only refreshed when the service allocated to the parameter switches from the offline to the operator or automatic state – otherwise, VOut remains constant. And thus a configuration parameter is above all suitable for specifying values which are required for the basic configuration of a service, and which are not permitted to change during the use of a service (For example, selection of the valve to be used for a dosing service).
-

20.1.2 Basic functions

20.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

20.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 120 Worst quality code enumeration

20.1.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

20.1.3 Functions

20.1.3.1 Service-parameter assignment

Fundamentally, there's an unequivocal service-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. When invoked, the parameter FB must also be informed of the associated service via the Service input.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
    Parameter1: AnaConfigParam := (TagName := 'Parameter1', TagDescription := '');
END_VAR

(* Implementation example ST *)
SUPER^();
Parameter1(Service := THIS^);
```

20.1.3.2 Input value (Operator/Internal/External)

Due to the fact that there are three input values, a decision must be made as to which value is active (VOp, VInt or VExt), and the value must be further processed for the calculation of output VOut. The source mode is available to this and, which controls both inputs. Upon initialisation, the manual source is active, i.e. VOp. The mode of operation and handling of the source mode is explained in a separate section. VOp and VExt are not inputs at the FB, and can only be set as an OPC UA variable. The selected scaling values (VMin and VMax) are valid for all inputs, as is VUnit.

20.1.3.3 Output value “VOut”

The selected input value is converted to output value “VOut” with the help of the standardisation values (“VMin” and “VMax”) and is read out from the FB. The standardisation values are available as inputs at the FB. VOut is only refreshed when the service allocated to the parameter switches from the offline to the operator or automatic state – otherwise, VOut remains constant.

20.1.3.4 Output “VRbk”

The “VRbk” output indicates the currently active input value, but without limiting the value to the standardisation values. “VRbk” is not updated when the parameter block is in the “Offline” state.

20.1.3.5 Output “VReq”

The “VRbk” output indicates the currently active input value, limited to the standardisation values. “VReq” is not updated when the parameter block is in the “Offline” state.

20.1.3.6 Unit of measure for “VOp”, “VInt”, “VExt” and “VOut”

A unit of measure can be entered for the input and output values via the “VUnit” input variable at the FB. The “UnitSettings” enumeration is used in order to specify a unit of measure. The unit of measure is only used for displays and has no influence on the conversion.

20.1.3.7 Monitoring the standardisation values

As described above, the standardisation values limit the input and output values and are used for conversion. It must also be assured that the minimum standardisation values are not greater than the associated maximum standardisation values. Otherwise, there’s no guarantee that the conversion and the graphics in the visualisation will include sensible values.

20.1.3.8 Scaling values for the visualisation

One Min and one Max variable (VScImin and VScIMax) are used to set the display limits of each visual representation of the output value. These variables are used exclusively for the representation of graphic elements in the visualisation.

20.1.4 Source mode

20.1.4.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (external). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (VInt) the “SrcIntAct” output is set to TRUE. When the manual source is active (VExt), the “SrcExtAct” output bit is set to TRUE.

The internal source is active as a default setting.

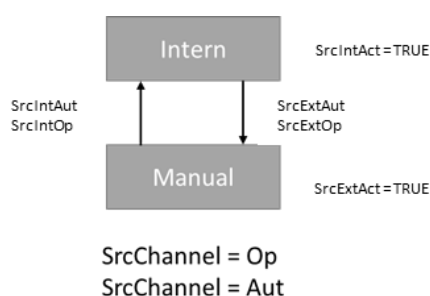


Fig. 143 Statemachine source mode

20.1.4.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcExtAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAut” and “SrcExtAut” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcExtAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

20.1.4.3 Module-external access (OPC UA)

The “SrcIntOp” and “SrcExtOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “SrcIntOp” (internal source) or “SrcExtOp” (external source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcExtOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntOp” and “SrcExtOp”) are set simultaneously, “SrcIntOp” has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “SrcIntOp”, “SrcExtOp”, “SrcIntAut” or “SrcExtAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

20.1.5 Operation mode

20.1.5.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode.

The “StateChannel” input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the “StateOffAct” output is set to TRUE. When the operator mode is active “StateOpAct” is set to TRUE, and when the automatic mode is active “StateAutAct” is set to TRUE.

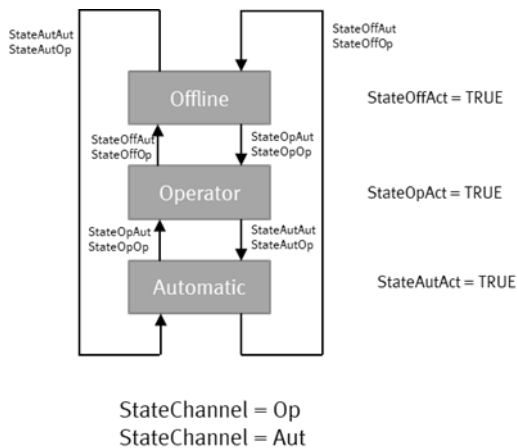


Fig. 144 Statemachine operation mode

20.1.5.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- “StateChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “StateOffAut” input (switch to offline mode), the “StateOpAut” input (switch to operator mode) or the “StateAutAut” input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the source is changed to or remains in the offline mode. Since it’s not possible to

switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

20.1.5.3 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.
- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

20.1.5.4 Synchronisation of the operation and source modes

As described above, the parameter block also has its own state (Offline/Operator/Automatic) and permits value specifications from various sources (Operator/Internal/External). Due to the fact that the parameter is unequivocally assigned to a certain service, in many cases it makes good sense to synchronise the state of the service and the parameter, as well as their channel for value specification. The Sync input can be used to this end. If this input is set to True (default value), the parameter’s own operation and source modes (see below) are ignored, and the current operation and source modes of the assigned service are used instead.

Note:

If the assigned service enters the offline state while synchronisation is activated, the parameter remains in its most recent state. If the parameter should also be set to the offline state, this transition must be triggered separately.

20.2 AnaProcedureParam

20.2.1 Brief description

The AnaProcedureParam function block (FB) can be used to define an analogue procedure parameter (floating-point number) for a service procedure. The parameter's current value is available at the VOut output. As opposed to a simple value specification, For example, when using a VAR_INPUT of the REAL type, this provides for the following extensions:

- Analogous to a service function block, this function block also has its own state (Offline/Operator/Automatic) and permits value specifications from various sources (Operator/Internal/External).
- VOut is only refreshed when the service allocated to the parameter is started with the procedure assigned to the parameter (i.e. its Starting state is executed) – otherwise, VOut remains constant. And thus a procedure parameter is above all suitable for specifying values which are required for a certain procedure of a service, and which are not permitted to change during execution of the service (For example, specification of a duration for a time-limited dosing procedure).

20.2.2 Basic functions

20.2.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

20.2.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 121 Worst quality code enumeration

20.2.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

20.2.3 Functions

20.2.3.1 Service-procedure-parameter assignment

Fundamentally, there's an unequivocal service-procedure-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. Furthermore, the procedure to which the parameter is assigned must be defined via the ProcedureID parameter during initialisation. When invoked, the parameter FB must also be informed of the associated service via the Service input.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
    Timelimited: ServiceProcedure := (TagName := 'Time-limited', TagDescription := '', ProcedureID := 1,
    IsSelfCompleting := TRUE);
    Time: AnaProcedureParam := (TagName := 'Time', TagDescription := '', ProcedureID := 1);
END_VAR

(* Implementation example ST *)
SUPER^();
Continuously (Service := THIS^);
Time(Service := THIS^);
```

20.2.3.2 Input value (Operator/Internal/External)

Due to the fact that there are three input values, a decision must be made as to which value is active (VOp, VInt or VExt), and the value must be further processed for the calculation of output VOut. The source mode is available to this and, which controls both inputs. Upon initialisation, the manual source is active, i.e. VOp. The mode of operation and handling of the source mode is explained in a separate section. VOp and VExt are not inputs at the FB, and can only be set as an OPC UA variable. The selected scaling values (VMin and VMax) are valid for all inputs, as is VUnit.

20.2.3.3 Output value “VOut”

The selected input value is converted to output value “VOut” with the help of the standardisation values (“VMin” and “VMax”) and is read out from the FB. The standardisation values are available as inputs at the FB. VOut is only refreshed when the service allocated to the parameter is started with the procedure assigned to the parameter (i.e. its Starting state is executed) – otherwise, VOut remains constant.

20.2.3.4 Output “VRbk”

The “VRbk” output indicates the currently active input value, but without limiting the value to the standardisation values. “VRbk” is not updated when the parameter block is in the “Offline” state.

20.2.3.5 Output “VReq”

The “VRbk” output indicates the currently active input value, limited to the standardisation values. “VReq” is not updated when the parameter block is in the “Offline” state.

20.2.3.6 Unit of measure for “VOp”, “VInt”, “VExt” and “VOut”

A unit of measure can be entered for the input and output values via the “VUnit” input variable at the FB. The “UnitSettings” enumeration is used in order to specify a unit of measure. The unit of measure is only used for displays and has no influence on the conversion.

20.2.3.7 Monitoring the standardisation values

As described above, the standardisation values limit the input and output values and are used for conversion. It must also be assured that the minimum standardisation values are not greater than the associated maximum standardisation values. Otherwise, there’s no guarantee that the conversion and the graphics in the visualisation will include sensible values.

20.2.3.8 Scaling values for the visualisation

One Min and one Max variable (VScImin and VScIMax) are used to set the display limits of each visual representation of the output value. These variables are used exclusively for the representation of graphic elements in the visualisation.

20.2.4 Source mode

20.2.4.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (external). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (VInt) the “SrcIntAct” output is set to TRUE. When the manual source is active (VExt), the “SrcExtAct” output bit is set to TRUE.

The internal source is active as a default setting.

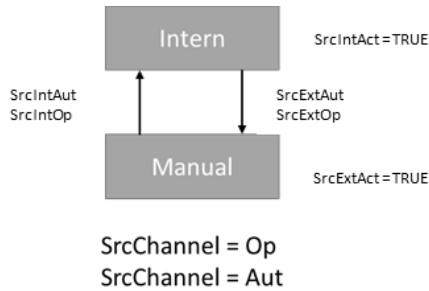


Fig. 145 Statemachine source mode

20.2.4.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcExtAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAut” and “SrcExtAut” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcExtAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

20.2.4.3 Module-external access (OPC UA)

The “SrcIntOp” and “SrcExtOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “SrcIntOp” (internal source) or “SrcExtOp” (external source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcExtOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op

variables, requests are not processed by the FB although they're acknowledged. Invalid requests are not saved for a later point in time.

If both inputs ("SrcIntOp" and "SrcExtOp") are set simultaneously, "SrcIntOp" has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable "SrcIntOp", "SrcExtOp", "SrcIntAut" or "SrcExtAut", the variable is set back to FALSE in order to acknowledge the request. For this reason, it's important to ensure that these inputs are not permanently assigned to the FB.

20.2.5 Operation mode

20.2.5.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode.

The "StateChannel" input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the "StateOffAct" output is set to TRUE. When the operator mode is active "StateOpAct" is set to TRUE, and when the automatic mode is active "StateAutAct" is set to TRUE.

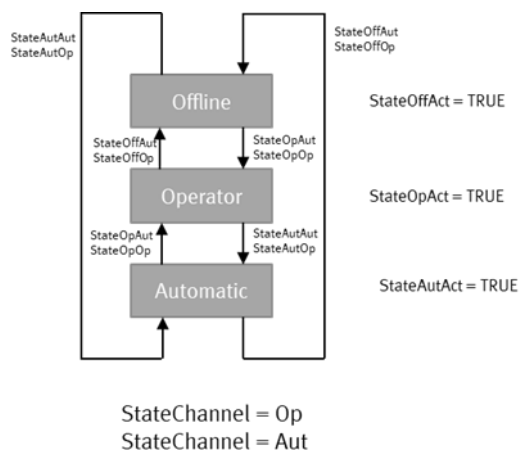


Fig. 146 Statemachine operation mode

20.2.5.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- “StateChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “StateOffAut” input (switch to offline mode), the “StateOpAut” input (switch to operator mode) or the “StateAutAut” input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the source is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

20.2.5.3 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.
- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

20.2.5.4 Synchronisation of the operation and source modes

As described above, the parameter block also has its own state (Offline/Operator/Automatic) and permits value specifications from various sources (Operator/Internal/External). Due to the fact that the parameter is unequivocally assigned to a certain service, in many cases it makes good sense to synchronise the state of the service and the parameter, as well as their channel for value specification. The Sync input can be used to this end. If this input is set to True (default value), the parameter’s own operation and source modes (see below) are ignored, and the current operation and source modes of the assigned service are used instead.

Note:

If the assigned service enters the offline state while synchronisation is activated, the parameter remains in its most recent state. If the parameter should also be set to the offline state, this transition must be triggered separately.

20.3 AnaProcessValueIn

20.3.1 Brief description

The AnaProcessValueIn function block (FB) can be used for continuous transmission of an external, analogue value (floating-point number) to a service procedure. The parameter's current value is available at the VOut output.

20.3.2 Basic functions

20.3.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

20.3.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All WQCs are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 122 Worst quality code enumeration

20.3.3 Functions

20.3.3.1 Service-procedure-parameter assignment

Fundamentally, there's an unequivocal service-procedure-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. Furthermore, the procedure to which the parameter is assigned must be defined via the ProcedureID parameter during initialisation.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
    Timelimited: ServiceProcedure := (TagName := 'time-limited', TagDescription := '', ProcedureID := 1,
    IsSelfCompleting := TRUE);
    Level: AnaProcessValueIn := (TagName := 'Level', TagDescription := '', ProcedureID := 1);
END_VAR

(* Implementation example ST *)
SUPER^();
Timelimited(Service := THIS^);
Level();
```

20.3.3.2 Input value „VExt“

The input value is represented by the “VExt” variable and is made available as an OPC UA variable.

20.3.3.3 Output value “VOut“

The input value is forwarded to output value “VOut” without conversion and read out as an output at the FB.

20.3.3.4 Unit of measure for “VExt” and “VOut“

A unit of measure can be entered for the input and output values via the “VUnit” input variable at the FB. The “UnitSettings” enumeration is used in order to specify a unit of measure. The unit of measure is only used for displays and has no influence on the conversion.

20.3.3.5 Scaling values for the visualisation

One Min and one Max variable (VScIMin and VScIMax) are used to set the display limits of each visual representation of the output value. These variables are used exclusively for the representation of graphic elements in the visualisation.

20.4 AnaProcessValueOut

20.4.1 Brief description

The AnaProcessValueOut function block (FB) is used for continuous feedback of an analogue value (floating-point number) from a service procedure.

20.4.2 Basic functions

20.4.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

20.4.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 123 Worst quality code enumeration

20.4.3 Functions

20.4.3.1 Service-procedure-parameter assignment

Fundamentally, there's an unequivocal service-procedure-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. Furthermore, the procedure to which the parameter is assigned must be defined via the ProcedureID parameter during initialisation.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
    Timelimited: ServiceProcedure := (TagName := 'time-limited', TagDescription := '', ProcedureID := 1,
    IsSelfCompleting := TRUE);
    Level: AnaProcessValueOut := (TagName := 'Level', TagDescription := '', ProcedureID := 1);
END_VAR

(* Implementation example ST *)
SUPER^();
Timelimited(Service := THIS^);
Level();
```

20.4.3.2 Input values “FV” and “VSim” (field value or simulated value)

The input value is represented by the “FV” variable and is made available to the FB as an input. This input can be linked to a variable or an address of an analogue input card. Scaling of the input value is adjusted via standardisation values “FVNormMin” and “FVNormMax”. Both of these values are also inputs at the FB.

If no input value is available, a value can be simulated at the FB via the “VSim” input. The simulation is activated via the “VSimEn” input bit (= TRUE). Output value “V” is simulated here directly. This means that the simulated value is not scaled. It's only checked to determine if the simulated values fall short of the “VNormMin” value or exceed the “VNormMax” value. If the limit values are not complied with, the corresponding limit value is read out as “V”.

20.4.3.3 Output value “V”

Input value “FV” is converted to output value “V” with the help of the standardisation values (“FVNormMin”, “FVNormMax”, “VNormMin” and “VNormMax”) and is read out from the FB. As is also the case for the standardisation values for the input value, the standardisation values for “V” are available as inputs at the FB. If “V” exceeds or falls short of its standardisation values (“VNormMin” and “VNormMax”), the corresponding standardisation value is read out as an output.

If the simulation is used, “Simulated Value” is set as the WQC. The simulated value is not scaled for read-out at output “V”. This means that only limit value monitoring takes place, but no conversion.

20.4.3.4 Conversion of input value “FV” to output value “V”

The standardisation values, which can be read into the FB as inputs, are required for the input as well as for the output in order to convert the input value to an output value.

The conversion formula is as follows:

$$V = \frac{VNormMax - VNormMin}{FVNormMax - FVNormMin} * (FV - FVNormMin) + VNormMin$$

Input: FVNormMin and FVNormMax

Output: VNormMin and VNormMax

20.4.3.5 Unit of measure for “V”

A unit of measure can be entered for the output value via the “VUnit” input variable at the FB. The “UnitSettings” enumeration is used in order to specify a unit of measure. The unit of measure is only used for displays and has no influence on the conversion.

20.4.3.6 Monitoring the standardisation values

As described above, the standardisation values limit the input and output values and are used for conversion. It must also be assured that the minimum standardisation values are not greater than the associated maximum standardisation values. Otherwise, there’s no guarantee that the conversion and the graphics in the visualisation will include sensible values.

Furthermore, the difference between “FVNormMax” and “FVNormMin” must not be zero, because it’s not possible to divide by zero. In this case zero is read out at output value “V”.

20.4.3.7 Scaling values for the visualisation

One Min and one Max variable (VScIMin and VScIMax) are used to set the display limits of each visual representation of the output value. These variables are used exclusively for the representation of graphic elements in the visualisation. If output “FullScIEn” is set to TRUE, standardisation values “VNormMin” and “VNormMax” are used automatically as display limits. If the display limits are set incorrectly, they’re set automatically to the standardisation values. Refer to the section entitled “FB-specific worst quality code” in this regard.

20.4.4 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
				VScIMax < VScImin ⇒ VScIMax = VNormMax ⇒ VScImin = VNormMin
				VScIMax < VNormMin or VScIMax > VNormMax ⇒ VScIMax = VNormMax
				VScImin < VNormMin or VScImin > VNormMax ⇒ VScImin = VNormMin
				VScIMax – VScImin = 0 ⇒ VScIMax = VNormMax ⇒ VScImin = VNormMin
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 124 AnaProcessValueOut specific worst quality code

20.4.5 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It’s even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
VNormMax	REAL	100	High Scale Value of Output Value (High limit of Output Value and scaling factor for V calculation)
VNormMin	REAL	0	Low Scale Value of Output Value (Low limit of Output Value and scaling factor for V calculation)

Tab. 125 AnaProcessValueOut persistent variables

20.5 AnaReportValue

20.5.1 Brief description

The AnaReportValue function block (FB) is used to make an analogue value (floating-point number) available from a service procedure for the purpose of documentation or verification. The FB extends the functionality of an AnaProcessValueOut such that the value of the parameter (“V”) is frozen if the associated service is in the completed, stopped or aborted state. This makes it possible for an interconnected documentation system to read out the parameter’s value in these states, e.g. for the purpose of verification.

20.5.2 Basic functions

20.5.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they’re statically processed, both values must be entered in the following format: “Tagname”.

20.5.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled “FB-specific worst quality code”.

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 126 Worst quality code enumeration

20.5.3 Functions

20.5.3.1 Service-procedure-parameter assignment

Fundamentally, there's an unequivocal service-procedure-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. Furthermore, the procedure to which the parameter is assigned must be defined via the ProcedureID parameter during initialisation. When invoked, the parameter FB must also be informed of the associated service via the Service input.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
    Timelimited: ServiceProcedure := (TagName := 'time-limited', TagDescription := '', ProcedureID := 1,
    IsSelfCompleting := TRUE);
    Level: AnaReportValue := (TagName := 'Level', TagDescription := '', ProcedureID := 1);
END_VAR

(* Implementation example ST *)
SUPER^();
Timelimited(Service := THIS^);
Level(Service := THIS^);
```

20.5.3.2 Input values “FV” and “VSim” (field value or simulated value)

The input value is represented by the “FV” variable and is made available to the FB as an input. This input can be linked to a variable or an address of an analogue input card. Scaling of the input value is adjusted via standardisation values “FVNormMin” and “FVNormMax”. Both of these values are also inputs at the FB. If no input value is available, a value can be simulated at the FB via the “VSim” input. The simulation is activated via the “VSimEn” input bit (= TRUE). Output value “V” is simulated here directly. This means that the simulated value is not scaled. It's only checked to determine if the simulated values fall short of the “VNormMin” value or exceed the “VNormMax” value. If the limit values are not complied with, the corresponding limit value is read out as “V”.

20.5.3.3 Output value “V”

Input value “FV” is converted to output value “V” with the help of the standardisation values (“FVNormMin”, “FVNormMax”, “VNormMin” and “VNormMax”) and is read out from the FB. As is also the case for the standardisation values for the input value, the standardisation values for “V” are available as inputs at the FB. If “V” exceeds or falls short of its standardisation values (“VNormMin” and “VNormMax”), the corresponding standardisation value is read out as an output.

If the simulation is used, “Simulated Value” is set as the WQC. The simulated value is not scaled for read-out at output “V”. This means that only limit value monitoring takes place, but no conversion.

Input value (“V”) is frozen if the associated service is in the completed, stopped or aborted state. If the service is in the offline state, the value is set to 0.

20.5.3.4 Conversion of input value “FV” to output value “V”

The standardisation values, which can be read into the FB as inputs, are required for the input as well as for the output in order to convert the input value to an output value.

The conversion formula is as follows:

$$V = \frac{VNormMax - VNormMin}{FVNormMax - FVNormMin} * (FV - FVNormMin) + VNormMin$$

Input: FVNormMin and FVNormMax

Output: VNormMin and VNormMax

20.5.3.5 Unit of measure for “V”

A unit of measure can be entered for the output value via the “VUnit” input variable at the FB. The “UnitSettings” enumeration is used in order to specify a unit of measure. The unit of measure is only used for displays and has no influence on the conversion.

20.5.3.6 Monitoring the standardisation values

As described above, the standardisation values limit the input and output values and are used for conversion. It must also be assured that the minimum standardisation values are not greater than the associated maximum standardisation values. Otherwise, there’s no guarantee that the conversion and the graphics in the visualisation will include sensible values.

Furthermore, the difference between “FVNormMax” and “FVNormMin” must not be zero, because it’s not possible to divide by zero. In this case zero is read out at output value “V”.

20.5.3.7 Scaling values for the visualisation

One Min and one Max variable (VScIMin and VScIMax) are used to set the display limits of each visual representation of the output value. These variables are used exclusively for the representation of graphic elements in the visualisation. If output “FullScIEn” is set to TRUE, standardisation values “VNormMin” and “VNormMax” are used automatically as display limits. If the display limits are set incorrectly, they’re set automatically to the standardisation values. Refer to the section entitled “FB-specific worst quality code” in this regard.

20.5.4 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
				VScIMax < VScImin ⇒ VScIMax = VNormMax ⇒ VScImin = VNormMin
				VScIMax < VNormMin or VScIMax > VNormMax ⇒ VScIMax = VNormMax
				VScImin < VNormMin or VScImin > VNormMax ⇒ VScImin = VNormMin
				VScIMax – VScImin = 0 ⇒ VScIMax = VNormMax ⇒ VScImin = VNormMin
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 127 AnaReportValue specific worst quality code

20.5.5 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It’s even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
VNormMax	REAL	100	High Scale Value of Output Value (High limit of Output Value and scaling factor for V calculation)
VNormMin	REAL	0	Low Scale Value of Output Value (Low limit of Output Value and scaling factor for V calculation)

Tab. 128 AnaReportValue persistent variables

21 Binary service parameters

21.1 BinConfigParam

21.1.1 Brief description

The BinConfigParam function block (FB) can be used to define a binary (Boolean) configuration parameter for a service. The parameter's current value is available at the VOut output. As opposed to a simple value specification, For example, when using a VAR_INPUT of the BOOL type, this provides for the following extensions:

- Analogous to a service function block, this function block also has its own state (Offline/Operator/Automatic) and permits value specifications from various sources (Operator/Internal/External).
- VOut is only refreshed when the service allocated to the parameter switches from the offline to the operator or automatic state – otherwise, VOut remains constant. And thus a configuration parameter is above all suitable for specifying values which are required for the basic configuration of a service, and which are not permitted to change during the use of a service (For example, selection of the valve to be used for a dosing service).

21.1.2 Basic functions

21.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

21.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 129 Worst quality code enumeration

21.1.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

21.1.3 Functions

21.1.3.1 Service-parameter assignment

Fundamentally, there's an unequivocal service-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. When invoked, the parameter FB must also be informed of the associated service via the Service input.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
    Parameter1: BinConfigParam := (TagName := 'Parameter1', TagDescription := '');
END_VAR

(* Implementation example ST *)
SUPER^();
Parameter1(Service := THIS^);
```

21.1.3.2 Input value (Operator/Internal/External)

Due to the fact that there are three input values, a decision must be made as to which value is active (VOp, VInt or VExt), and the value must be further processed for the calculation of output VOut. The source mode is available to this and, which controls both inputs. Upon initialisation, the manual source is active, i.e. VOp. The mode of operation and handling of the source mode is explained in a separate section. VOp and VExt are not inputs at the FB and can only be set as an OPC UA variable.

21.1.3.3 Output value “VOut”

The selected input value is read out as “VOut” as an output at the FB. VOut is only refreshed when the service allocated to the parameter switches from the offline to the operator or automatic state – otherwise, VOut remains constant.

21.1.3.4 Output “VRbk”

Der Ausgang „VRbk“ gibt den aktuell aktiven Eingangswert an, jedoch ohne den Wert auf die Normierungswerte zu begrenzen. „VRbk“ wird nicht aktualisiert, wenn sich der Parameterbaustein im Zustand „Offline“ befindet.

21.1.3.5 Output „VReq“

The “VRbk” output indicates the currently active input value, limited to the standardisation values. “VReq” is not updated when the parameter block is in the “Offline” state.

21.1.4 Source mode

21.1.4.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (external). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (VInt) the “SrcIntAct” output is set to TRUE. When the manual source is active (VExt), the “SrcExtAct” output bit is set to TRUE.

The internal source is active as a default setting.

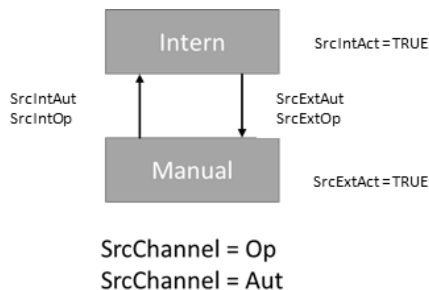


Fig. 147 Statemachine source mode

21.1.4.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutoOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcExtAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAct” and “SrcExtAct” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcExtAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

21.1.4.3 Module-external access (OPC UA)

The “SrcIntOp” and “SrcExtOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “SrcIntOp” (internal source) or “SrcExtOp” (external source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcExtOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntOp” and “SrcExtOp”) are set simultaneously, “SrcIntOp” has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “SrcIntOp”, “SrcExtOp”, “SrcIntAut” or “SrcExtAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

21.1.5 Operation mode

21.1.5.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode.

The “StateChannel” input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the “StateOffAct” output is set to TRUE. When the operator mode is active “StateOpAct” is set to TRUE, and when the automatic mode is active “StateAutAct” is set to TRUE.

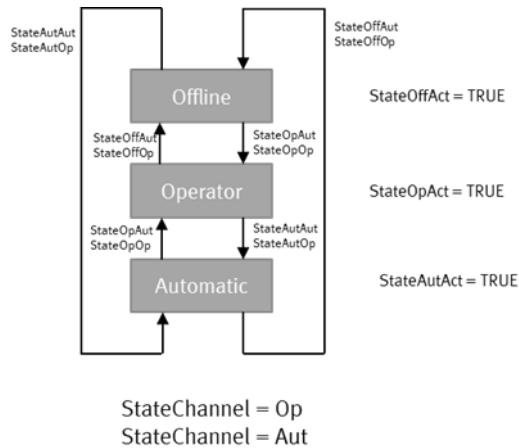


Fig. 148 Statemachine operation mode

21.1.5.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- “StateChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “StateOffAut” input (switch to offline mode), the “StateOpAut” input (switch to operator mode) or the “StateAutAut” input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the source is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

21.1.5.3 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.
- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.

- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn't necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they're acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it's not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it's important to ensure that these inputs are not permanently assigned to the FB.

21.1.5.4 Synchronisation of the operation and source modes

As described above, the parameter block also has its own state (Offline/Operator/Automatic) and permits value specifications from various sources (Operator/Internal/External). Due to the fact that the parameter is unequivocally assigned to a certain service, in many cases it makes good sense to synchronise the state of the service and the parameter, as well as their channel for value specification. The Sync input can be used to this end. If this input is set to True (default value), the parameter's own operation and source modes (see below) are ignored, and the current operation and source modes of the assigned service are used instead.

Note:

If the assigned service enters the offline state while synchronisation is activated, the parameter remains in its most recent state. If the parameter should also be set to the offline state, this transition must be triggered separately.

21.2 BinProcedureParam

21.2.1 Brief description

The BinProcedureParam function block (FB) can be used to define a binary (Boolean) procedure parameter for a service procedure. The parameter's current value is available at the VOut output. As opposed to a simple value specification, For example, when using a VAR_INPUT of the BOOL type, this provides for the following extensions:

- Analogous to a service function block, this function block also has its own state (Offline/Operator/Automatic) and permits value specifications from various sources (Operator/Internal/External).
- VOut is only refreshed when the service allocated to the parameter is started with the procedure assigned to the parameter (i.e. its Starting state is executed) – otherwise, VOut remains constant. And thus a procedure parameter is above all suitable for specifying values which are required for a certain procedure of a service, and which are not permitted to change during execution of the service (For example, specification of a duration for a time-limited dosing procedure).

21.2.2 Basic functions

21.2.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

21.2.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 130 Worst quality code enumeration

21.2.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

21.2.3 Functions

21.2.3.1 Service-procedure-parameter assignment

Fundamentally, there's an unequivocal service-procedure-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. Furthermore, the procedure to which the parameter is assigned must be defined via the ProcedureID parameter during initialisation. When invoked, the parameter FB must also be informed of the associated service via the Service input.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
    Timelimited: ServiceProcedure := (TagName := 'time-limited', TagDescription := '', ProcedureID := 1,
    IsSelfCompleting := TRUE);
    Time: BinProcedureParam := (TagName := 'Time', TagDescription := '', ProcedureID := 1);
END_VAR

(* Implementation example ST *)
SUPER^();
Timelimited(Service := THIS^);
Time(Service := THIS^);
```

21.2.3.2 Input value (Operator/Internal/External)

Due to the fact that there are three input values, a decision must be made as to which value is active (VOp, VInt or VExt), and the value must be further processed for the calculation of output VOut. The source mode is available to this and, which controls both inputs. Upon initialisation, the manual source is active, i.e. VOp. The mode of operation and handling of the source mode is explained in a separate section. VOp and VExt are not inputs at the FB, and can only be set as an OPC UA variable.

21.2.3.3 Output value “VOut”

The selected input value is read out as “VOut” as an output at the FB. VOut is only refreshed when the service allocated to the parameter is started with the procedure assigned to the parameter (i.e. its Starting state is executed) – otherwise, VOut remains constant.

21.2.3.4 Output “VRbk”

The “VRbk” output indicates the currently active input value. “VRbk” is not updated when the parameter block is in the “Offline” state.

21.2.3.5 Output „VReq“

The “VRbk” output indicates the currently active input value. “VReq” is not updated when the parameter block is in the “Offline” state.

21.2.4 Source mode

21.2.4.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (external). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (VInt) the “SrcIntAct” output is set to TRUE. When the manual source is active (VExt), the “SrcExtAct” output bit is set to TRUE.

The internal source is active as a default setting.

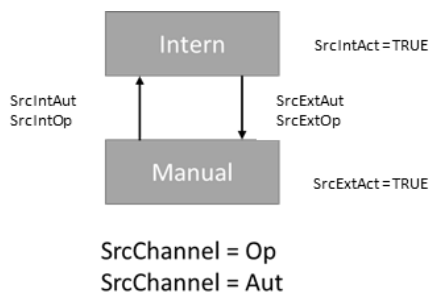


Fig. 149 Statemachine source mode

21.2.4.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcExtAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAut” and “SrcExtAut” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcExtAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

21.2.4.3 Module-external access (OPC UA)

The “SrcIntOp” and “SrcExtOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “SrcIntOp” (internal source) or “SrcExtOp” (external source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcExtOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntOp” and “SrcExtOp”) are set simultaneously, “SrcIntOp” has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “SrcIntOp”, “SrcExtOp”, “SrcIntAut” or “SrcExtAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

21.2.5 Operation mode

21.2.5.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode.

The “StateChannel” input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the “StateOffAct” output is set to TRUE. When the operator mode is active “StateOpAct” is set to TRUE, and when the automatic mode is active “StateAutAct” is set to TRUE.

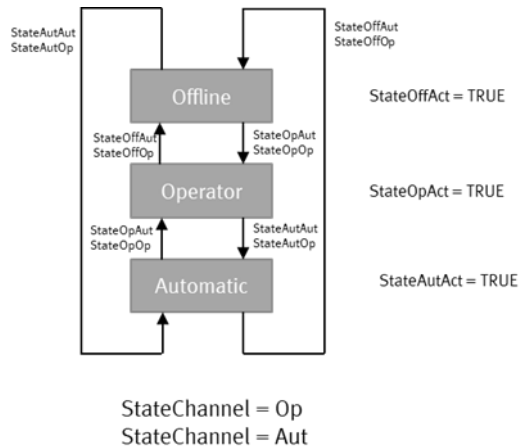


Fig. 150 Statemachine operation mode

21.2.5.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- “StateChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “StateOffAut” input (switch to offline mode), the “StateOpAut” input (switch to operator mode) or the “StateAutAut” input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the source is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

21.2.5.3 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.

- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

21.2.5.4 Synchronisation of the operation and source modes

As described above, the parameter block also has its own state (Offline/Operator/Automatic) and permits value specifications from various sources (Operator/Internal/External). Due to the fact that the parameter is unequivocally assigned to a certain service, in many cases it makes good sense to synchronise the state of the service and the parameter, as well as their channel for value specification. The Sync input can be used to this end. If this input is set to True (default value), the parameter’s own operation and source modes (see below) are ignored, and the current operation and source modes of the assigned service are used instead.

Note:

If the assigned service enters the offline state while synchronisation is activated, the parameter remains in its most recent state. If the parameter should also be set to the offline state, this transition must be triggered separately.

21.3 BinProcessValueIn

21.3.1 Brief description

The BinProcessValueIn function block (FB) can be used for continuous transmission of an external, binary (Boolean) value to a service procedure. The parameter's current value is available at the VOut output.

21.3.2 Basic functions

21.3.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

21.3.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 131 Worst quality code enumeration

21.3.3 Functions

21.3.3.1 Service-procedure-parameter assignment

Fundamentally, there's an unequivocal service-procedure-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. Furthermore, the procedure to which the parameter is assigned must be defined via the ProcedureID parameter during initialisation.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
    Timelimited: ServiceProcedure := (TagName := 'time-limited', TagDescription := '', ProcedureID := 1,
    IsSelfCompleting := TRUE);
    Level: BinProcessValueIn := (TagName := 'Level', TagDescription := '', ProcedureID := 1);
END_VAR

(* Implementation example ST *)
SUPER^0;
Timelimited(Service := THIS^);
Level(Service := THIS^);
```

21.3.3.2 Input value „VExt“

The input value is represented by the “VExt” variable and is made available as an OPC UA variable.

21.3.3.3 Output value “VOut“

The input value is forwarded to output value “VOut” and read out as an output at the FB.

21.3.3.4 Status text “VState0” and “VState1“

The “VState0” and “VState1” inputs are available at the FB as two separate string variables. Texts can be stored for both states at these inputs. These describe the states additionally in words. For example, the text “Lamp on” can be assigned to the VState1 input for the value TRUE.

24 characters are available for each status text.

21.4 BinProcessValueOut

21.4.1 Brief description

The BinProcessValueOut function block (FB) is used for continuous feedback of an binary (Boolean) value from a service procedure.

21.4.2 Basic functions

21.4.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

21.4.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 132 Worst quality code enumeration

21.4.3 Functions

21.4.3.1 Service-procedure-parameter assignment

Fundamentally, there's an unequivocal service-procedure-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. Furthermore, the procedure to which the parameter is assigned must be defined via the ProcedureID parameter during initialisation.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
    Timelimited: ServiceProcedure := (TagName := 'time-limited', TagDescription := '', ProcedureID := 1,
    IsSelfCompleting := TRUE);
    Level: BinProcessValueOut := (TagName := 'Level', TagDescription := '', ProcedureID := 1);
END_VAR

(* Implementation example ST *)
SUPER^0;
Timelimited(Service := THIS^);
Level(Service := THIS^);
```

21.4.3.2 Input values “FV” and “VSim” (field value or simulated value)

The input value is represented by the “FV” variable and is made available to the FB as an input. This input can be linked to a variable or an address of a digital input card.

If no input value is available, a value can be simulated at the FB via the “VSim” input. The simulation is activated via the “VSimEn” input bit (= TRUE). Output value “V” is simulated here directly.

21.4.3.3 Status text “VState0” and “VState1”

The “State0” and “State1” inputs are available at the FB as two separate string variables. Texts can be stored for both states at these inputs. These describe the states additionally in words. For example, the text “Tank is full” can be assigned to the VState1 input for the value TRUE.

24 characters are available for each status text.

21.4.4 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 133 BinProcessValueOut specific worst quality code

21.5 BinReportValue

21.5.1 Brief description

The BinReportValue function block (FB) is used to make a binary (Boolean) value available from a service procedure for the purpose of documentation or verification. The FB extends the functionality of a BinProcessValueOut such that the value of the parameter (“V”) is frozen if the associated service is in the completed, stopped or aborted state. This makes it possible for an interconnected documentation system to read out the parameter’s value in these states, e.g. for the purpose of verification.

21.5.2 Basic functions

21.5.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they’re statically processed, both values must be entered in the following format: “Tagname”.

21.5.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled “FB-specific worst quality code”.

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 134 Worst quality code enumeration

21.5.3 Functions

21.5.3.1 Service-procedure-parameter assignment

Fundamentally, there's an unequivocal service-procedure-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. Furthermore, the procedure to which the parameter is assigned must be defined via the ProcedureID parameter during initialisation. When invoked, the parameter FB must also be informed of the associated service via the Service input.

```
(* Declaration example *)  
FUNCTION_BLOCK Stir EXTENDS ServiceControl  
VAR  
    Timelimited: ServiceProcedure := (TagName := 'time-limited', TagDescription := '', ProcedureID := 1,  
    IsSelfCompleting := TRUE);  
    Level: BinReportValue := (TagName := 'Level', TagDescription := '', ProcedureID := 1);  
END_VAR  
  
(* Implementation example ST *)  
SUPER^();  
Timelimited(Service := THIS^);  
Level(Service := THIS^);
```

21.5.3.2 Input values “FV” and “VSim” (field value or simulated value)

The input value is represented by the “FV” variable and is made available to the FB as an input. This input can be linked to a variable or an address of a digital input card.

If no input value is available, a value can be simulated at the FB via the “VSim” input. The simulation is activated via the “VSimEn” input bit (= TRUE). Output value “V” is simulated here directly.

21.5.3.3 Status text “VState0” and “VState1”

The “State0” and “State1” inputs are available at the FB as two separate string variables. Texts can be stored for both states at these inputs. These describe the states additionally in words. For example, the text “Tank is full” can be assigned to the VState1 input for the value TRUE.

24 characters are available for each status text.

21.5.3.4 Output value “V”

Input value (“V”) is frozen if the associated service is in the completed, stopped or aborted state. If the service is in the offline state, the value is set to FALSE.

21.5.4 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 135 BinReportValue specific worst quality code

22 Digital service parameters

22.1 DIntConfigParam

22.1.1 Brief description

The DIntConfigParam function block (FB) can be used to define an analogue configuration parameter (double integer) for a service. The parameter's current value is available at the VOut output. As opposed to a simple value specification, For example, when using a VAR_INPUT of the DINT type, this provides for the following extensions:

- Analogous to a service function block, this function block also has its own state (Offline/Operator/Automatic) and permits value specifications from various sources (Operator/Internal/External).
- VOut is only refreshed when the service allocated to the parameter switches from the offline to the operator or automatic state – otherwise, VOut remains constant. And thus a configuration parameter is above all suitable for specifying values which are required for the basic configuration of a service, and which are not permitted to change during the use of a service (For example, selection of the valve to be used for a dosing service).
-

22.1.2 Basic functions

22.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

22.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 136 Worst quality code enumeration

22.1.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

22.1.3 Functions

22.1.3.1 Service-parameter assignment

Fundamentally, there's an unequivocal service-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. When invoked, the parameter FB must also be informed of the associated service via the Service input.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
    Parameter1: DIntConfigParam := (TagName := 'Parameter1', TagDescription := '');
END_VAR

(* Implementation example ST *)
SUPER^();
Parameter1(Service := THIS^);
```

22.1.3.2 Input value (Operator/Internal/External)

Due to the fact that there are three input values, a decision must be made as to which value is active (VOp, VInt or VExt), and the value must be further processed for the calculation of output VOut. The source mode is available to this and, which controls both inputs. Upon initialisation, the manual source is active, i.e. VOp. The mode of operation and handling of the source mode is explained in a separate section. VOp and VExt are not inputs at the FB, and can only be set as an OPC UA variable. The selected scaling values (VMin and VMax) are valid for all inputs, as is VUnit.

22.1.3.3 Output value “VOut”

The selected input value is converted to output value “VOut” with the help of the standardisation values (“VMin” and “VMax”) and is read out from the FB. The standardisation values are available as inputs at the FB. VOut is only refreshed when the service allocated to the parameter switches from the offline to the operator or automatic state – otherwise, VOut remains constant.

22.1.3.4 Output “VRbk”

The “VRbk” output indicates the currently active input value, but without limiting the value to the standardisation values. “VRbk” is not updated when the parameter block is in the “Offline” state.

22.1.3.5 Output „VReq“

The “VRbk” output indicates the currently active input value, limited to the standardisation values. “VReq” is not updated when the parameter block is in the “Offline” state.

22.1.3.6 Unit of measure for “VOp”, “VInt”, “VExt” and “VOut“

A unit of measure can be entered for the input and output values via the “VUnit” input variable at the FB. The “UnitSettings” enumeration is used in order to specify a unit of measure. The unit of measure is only used for displays and has no influence on the conversion.

22.1.3.7 Monitoring the standardisation values

As described above, the standardisation values limit the input and output values and are used for conversion. It must also be assured that the minimum standardisation values are not greater than the associated maximum standardisation values. Otherwise, there’s no guarantee that the conversion and the graphics in the visualisation will include sensible values.

22.1.3.8 Scaling values for the visualisation

One Min and one Max variable (VSclMin and VSclMax) are used to set the display limits of each visual representation of the output value. These variables are used exclusively for the representation of graphic elements in the visualisation.

22.1.4 Source mode

22.1.4.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (external). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (VInt) the “SrcIntAct” output is set to TRUE. When the manual source is active (VExt), the “SrcExtAct” output bit is set to TRUE.

The internal source is active as a default setting.

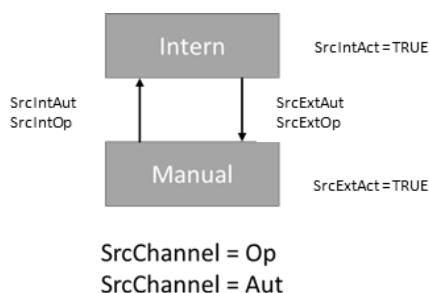


Fig. 151 Statemachine source mode

22.1.4.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcExtAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAut” and “SrcExtAut” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcExtAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

22.1.4.3 Module-external access (OPC UA)

The “SrcIntOp” and “SrcExtOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “SrcIntOp” (internal source) or “SrcExtOp” (external source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcExtOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntOp” and “SrcExtOp”) are set simultaneously, “SrcIntOp” has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “SrcIntOp”, “SrcExtOp”, “SrcIntAut” or “SrcExtAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

22.1.5 Operation mode

22.1.5.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode.

The “StateChannel” input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the “StateOffAct” output is set to TRUE. When the operator mode is active “StateOpAct” is set to TRUE, and when the automatic mode is active “StateAutAct” is set to TRUE.

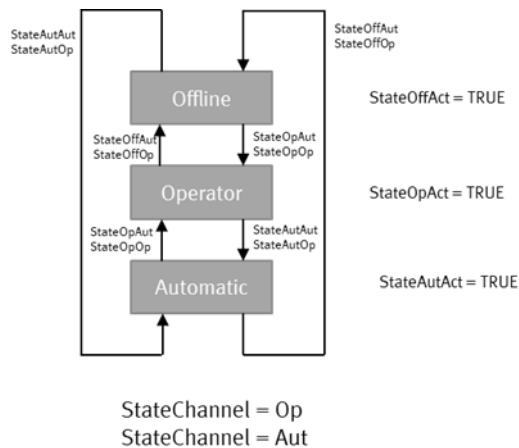


Fig. 152 Statemachine operation mode

22.1.5.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- “StateChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “StateOffAut” input (switch to offline mode), the “StateOpAut” input (switch to operator mode) or the “StateAutAut” input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the source is changed to or remains in the offline mode. Since it’s not possible to

switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

22.1.5.3 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.
- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

22.1.5.4 Synchronisation of the operation and source modes

As described above, the parameter block also has its own state (Offline/Operator/Automatic) and permits value specifications from various sources (Operator/Internal/External). Due to the fact that the parameter is unequivocally assigned to a certain service, in many cases it makes good sense to synchronise the state of the service and the parameter, as well as their channel for value specification. The Sync input can be used to this end. If this input is set to True (default value), the parameter’s own operation and source modes (see below) are ignored, and the current operation and source modes of the assigned service are used instead.

Note:

If the assigned service enters the offline state while synchronisation is activated, the parameter remains in its most recent state. If the parameter should also be set to the offline state, this transition must be triggered separately.

22.2 DIntProcedureParam

22.2.1 Brief description

The DIntProcedureParam function block (FB) can be used to define an analogue procedure parameter (double integer) for a service procedure. The parameter's current value is available at the VOut output. As opposed to a simple value specification, For example, when using a VAR_INPUT of the DINT type, this provides for the following extensions:

- Analogous to a service function block, this function block also has its own state (Offline/Operator/Automatic) and permits value specifications from various sources (Operator/Internal/External).
- VOut is only refreshed when the service allocated to the parameter is started with the procedure assigned to the parameter (i.e. its Starting state is executed) – otherwise, VOut remains constant. And thus a procedure parameter is above all suitable for specifying values which are required for a certain procedure of a service, and which are not permitted to change during execution of the service (For example, specification of a duration for a time-limited dosing procedure).
-

22.2.2 Basic functions

22.2.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

22.2.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 137 Worst quality code enumeration

22.2.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

22.2.3 Functions

22.2.3.1 Service-procedure- parameter assignment

Fundamentally, there's an unequivocal service-procedure-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. Furthermore, the procedure to which the parameter is assigned must be defined via the ProcedureID parameter during initialisation. When invoked, the parameter FB must also be informed of the associated service via the Service input.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
    Timelimited: ServiceProcedure := (TagName := 'time-limited', TagDescription := '', ProcedureID := 1,
    IsSelfCompleting := TRUE);
    Time: DIntProcedureParam := (TagName := 'Time', TagDescription := '', ProcedureID := 1);
END_VAR

(* Implementation example ST *)
SUPER^();
Timelimited(Service := THIS^);
Time(Service := THIS^);
```

22.2.3.2 Input value (Operator/Internal/External)

Due to the fact that there are three input values, a decision must be made as to which value is active (VOp, VInt or VExt), and the value must be further processed for the calculation of output VOut. The source mode is available to this and, which controls both inputs. Upon initialisation, the manual source is active, i.e. VOp. The mode of operation and handling of the source mode is explained in a separate section. VOp and VExt are not inputs at the FB, and can only be set as an OPC UA variable. The selected scaling values (VMin and VMax) are valid for all inputs, as is VUnit.

22.2.3.3 Output value “VOut“

The selected input value is converted to output value “VOut” with the help of the standardisation values (“VMin” and “VMax”) and is read out from the FB. The standardisation values are available as inputs at the FB. VOut is only refreshed when the service allocated to the parameter is started with the procedure assigned to the parameter (i.e. its Starting state is executed) – otherwise, VOut remains constant.

22.2.3.4 Output “VRbk”

The “VRbk” output indicates the currently active input value, but without limiting the value to the standardisation values. “VRbk” is not updated when the parameter block is in the “Offline” state.

22.2.3.5 Output „VReq“

The “VRbk” output indicates the currently active input value, limited to the standardisation values. “VReq” is not updated when the parameter block is in the “Offline” state.

22.2.3.6 Unit of measure for “VOp”, “VInt”, “VExt” and “VOut”

A unit of measure can be entered for the input and output values via the “VUnit” input variable at the FB. The “UnitSettings” enumeration is used in order to specify a unit of measure. The unit of measure is only used for displays and has no influence on the conversion.

22.2.3.7 Monitoring the standardisation values

As described above, the standardisation values limit the input and output values and are used for conversion. It must also be assured that the minimum standardisation values are not greater than the associated maximum standardisation values. Otherwise, there’s no guarantee that the conversion and the graphics in the visualisation will include sensible values.

22.2.3.8 Scaling values for the visualisation

One Min and one Max variable (VScIMin and VScIMax) are used to set the display limits of each visual representation of the output value. These variables are used exclusively for the representation of graphic elements in the visualisation.

22.2.4 Source mode

22.2.4.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (external). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (VInt) the “SrcIntAct” output is set to TRUE. When the manual source is active (VExt), the “SrcExtAct” output bit is set to TRUE.

The internal source is active as a default setting.

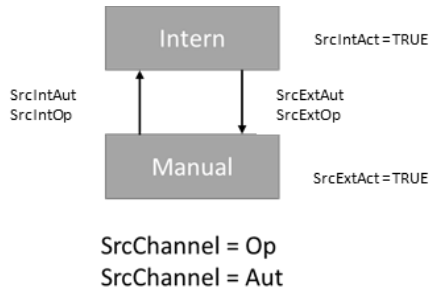


Fig. 153 Statemachine source mode

22.2.4.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcExtAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAut” and “SrcExtAut” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcExtAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

22.2.4.3 Module-external access (OPC UA)

The “SrcIntOp” and “SrcExtOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “SrcIntOp” (internal source) or “SrcExtOp” (external source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcExtOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op

variables, requests are not processed by the FB although they're acknowledged. Invalid requests are not saved for a later point in time.

If both inputs ("SrcIntOp" and "SrcExtOp") are set simultaneously, "SrcIntOp" has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable "SrcIntOp", "SrcExtOp", "SrcIntAut" or "SrcExtAut", the variable is set back to FALSE in order to acknowledge the request. For this reason, it's important to ensure that these inputs are not permanently assigned to the FB.

22.2.5 Operation mode

22.2.5.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode.

The "StateChannel" input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the "StateOffAct" output is set to TRUE. When the operator mode is active "StateOpAct" is set to TRUE, and when the automatic mode is active "StateAutAct" is set to TRUE.

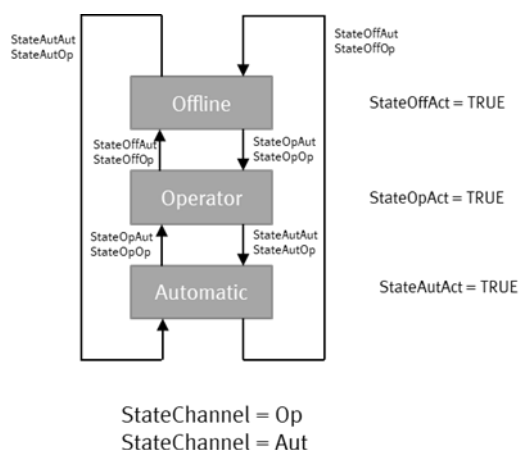


Fig. 154 Statemachine operation mode

22.2.5.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- "StateChannel" must be set to "Aut" via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit's faceplate).
- Set the "StateOffAut" input (switch to offline mode), the "StateOpAut" input (switch to operator mode) or the "StateAutAut" input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.

- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the source is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

22.2.5.3 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.
- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

22.2.5.4 Synchronisation of the operation and source modes

As described above, the parameter block also has its own state (Offline/Operator/Automatic) and permits value specifications from various sources (Operator/Internal/External). Due to the fact that the parameter is unequivocally assigned to a certain service, in many cases it makes good sense to synchronise the state of the service and the parameter, as well as their channel for value specification. The Sync input can be used to

this end. If this input is set to True (default value), the parameter's own operation and source modes (see below) are ignored, and the current operation and source modes of the assigned service are used instead.

Note:

If the assigned service enters the offline state while synchronisation is activated, the parameter remains in its most recent state. If the parameter should also be set to the offline state, this transition must be triggered separately.

22.3 DIntProcessValueIn

22.3.1 Brief description

The DIntProcessValueIn function block (FB) can be used for continuous transmission of an external, analogue value (double integer) to a service procedure. The parameter's current value is available at the VOut output.

22.3.2 Basic functions

22.3.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

22.3.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 138 Worst quality code enumeration

22.3.3 Functions

22.3.3.1 Service-procedure-parameter assignment

Fundamentally, there's an unequivocal service-procedure-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. Furthermore, the procedure to which the parameter is assigned must be defined via the ProcedureID parameter during initialisation.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
    Timelimited: ServiceProcedure := (TagName := 'time-limited', TagDescription := '', ProcedureID := 1,
    IsSelfCompleting := TRUE);
    Level: DIntProcessValueIn := (TagName := 'Level', TagDescription := '', ProcedureID := 1);
END_VAR

(* Implementation example ST *)
SUPER^0;
Timelimited(Service := THIS^);
Level(Service := THIS^);
```

22.3.3.2 Input value „VExt“

The input value is represented by the “VExt” variable and is made available as an OPC UA variable.

22.3.3.3 Output value “VOut“

The input value is forwarded to output value “VOut” without conversion and read out as an output at the FB.

22.3.3.4 Unit of measure for “VExt” and “VOut“

A unit of measure can be entered for the input and output values via the “VUnit” input variable at the FB. The “UnitSettings” enumeration is used in order to specify a unit of measure. The unit of measure is only used for displays and has no influence on the conversion.

22.3.3.5 Scaling values for the visualisation

One Min and one Max variable (VScImin and VScIMax) are used to set the display limits of each visual representation of the output value. These variables are used exclusively for the representation of graphic elements in the visualisation.

22.4 DIntProcessValueOut

22.4.1 Brief description

The DIntProcessValueOut function block (FB) is used for continuous feedback of an analogue value (double integer) from a service procedure.

22.4.2 Basic functions

22.4.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

22.4.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 139 Worst quality code enumeration

22.4.3 Functions

22.4.3.1 Service-procedure-parameter assignment

Fundamentally, there's an unequivocal service-procedure-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. Furthermore, the procedure to which the parameter is assigned must be defined via the ProcedureID parameter during initialisation.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
    Timelimited: ServiceProcedure := (TagName := 'time-limited', TagDescription := '', ProcedureID := 1,
    IsSelfCompleting := TRUE);
    Level: DIntProcessValueOut := (TagName := 'Level', TagDescription := '', ProcedureID := 1);
END_VAR

(* Implementation example ST *)
SUPER^0;
Timelimited(Service := THIS^);
Level(Service := THIS^);
```

22.4.3.2 Input values “FV” and “VSim” (field value or simulated value)

The input value is represented by the “FV” variable and is made available to the FB as an input. This input can be linked to a variable or an address of an analogue input card. Scaling of the input value is adjusted via standardisation values “FVNormMin” and “FVNormMax”. Both of these values are also inputs at the FB.

If no input value is available, a value can be simulated at the FB via the “VSim” input. The simulation is activated via the “VSimEn” input bit (= TRUE). Output value “V” is simulated here directly. This means that the simulated value is not scaled. It's only checked to determine if the simulated values fall short of the “VNormMin” value or exceed the “VNormMax” value. If the limit values are not complied with, the corresponding limit value is read out as “V”.

22.4.3.3 Output value “V”

Input value “FV” is converted to output value “V” with the help of the standardisation values (“FVNormMin”, “FVNormMax”, “VNormMin” and “VNormMax”) and is read out from the FB. As is also the case for the standardisation values for the input value, the standardisation values for “V” are available as inputs at the FB. If “V” exceeds or falls short of its standardisation values (“VNormMin” and “VNormMax”), the corresponding standardisation value is read out as an output.

If the simulation is used, “Simulated Value” is set as the WQC. The simulated value is not scaled for read-out at output “V”. This means that only limit value monitoring takes place, but no conversion.

22.4.3.4 Conversion of input value “FV” to output value “V”

The standardisation values, which can be read into the FB as inputs, are required for the input as well as for the output in order to convert the input value to an output value.

The conversion formula is as follows:

$$V = \frac{VNormMax - VNormMin}{FVNormMax - FVNormMin} * (FV - FVNormMin) + VNormMin$$

Input: FVNormMin and FVNormMax

Output: VNormMin and VNormMax

22.4.3.5 Unit of measure for “V”

A unit of measure can be entered for the output value via the “VUnit” input variable at the FB. The “UnitSettings” enumeration is used in order to specify a unit of measure. The unit of measure is only used for displays and has no influence on the conversion.

22.4.3.6 Monitoring the standardisation values

As described above, the standardisation values limit the input and output values and are used for conversion. It must also be assured that the minimum standardisation values are not greater than the associated maximum standardisation values. Otherwise, there’s no guarantee that the conversion and the graphics in the visualisation will include sensible values.

Furthermore, the difference between “FVNormMax” and “FVNormMin” must not be zero, because it’s not possible to divide by zero. In this case zero is read out at output value “V”.

22.4.3.7 Scaling values for the visualisation

One Min and one Max variable (VScIMin and VScIMax) are used to set the display limits of each visual representation of the output value. These variables are used exclusively for the representation of graphic elements in the visualisation. If output “FullScIEn” is set to TRUE, standardisation values “VNormMin” and “VNormMax” are used automatically as display limits. If the display limits are set incorrectly, they’re set automatically to the standardisation values. Refer to the section entitled “FB-specific worst quality code” in this regard.

22.4.4 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
				VScIMax < VScImin ⇒ VScIMax = VNormMax ⇒ VScImin = VNormMin
				VScIMax < VNormMin or VScIMax > VNormMax ⇒ VScIMax = VNormMax
				VScImin < VNormMin or VScImin > VNormMax ⇒ VScImin = VNormMin
				VScIMax – VScImin = 0 ⇒ VScIMax = VNormMax ⇒ VScImin = VNormMin
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 140 DIntProcessValueOut specific worst quality code

22.4.5 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It’s even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
VNormMax	DINT	100	High Scale Value of Output Value (High limit of Output Value and scaling factor for V calculation)
VNormMin	DINT	0	Low Scale Value of Output Value (Low limit of Output Value and scaling factor for V calculation)

Tab. 141 DIntProcessValueOut persistent variables

22.5 DIntReportValue

22.5.1 Brief description

The DIntReportValue function block (FB) is used to make an analogue value (double integer) available from a service procedure for the purpose of documentation or verification. The FB extends the functionality of a DIntProcessValueOut such that the value of the parameter ("V") is frozen if the associated service is in the completed, stopped or aborted state. This makes it possible for an interconnected documentation system to read out the parameter's value in these states, e.g. for the purpose of verification.

22.5.2 Basic functions

22.5.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

22.5.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 142 Worst quality code enumeration

22.5.3 Functions

22.5.3.1 Service-procedure-parameter assignment

Fundamentally, there's an unequivocal service-procedure-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. Furthermore, the procedure to which the parameter is assigned must be defined via the ProcedureID parameter during initialisation. When invoked, the parameter FB must also be informed of the associated service via the Service input.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
    Timelimited: ServiceProcedure := (TagName := 'time-limited', TagDescription := '', ProcedureID := 1,
    IsSelfCompleting := TRUE);
    Level: DIntReportValue := (TagName := 'Level', TagDescription := '', ProcedureID := 1);
END_VAR

(* Implementation example ST *)
SUPER^0;
Timelimited(Service := THIS^);
Level(Service := THIS^);
```

22.5.3.2 Input values “FV” and “VSim” (field value or simulated value)

The input value is represented by the “FV” variable and is made available to the FB as an input. This input can be linked to a variable or an address of an analogue input card. Scaling of the input value is adjusted via standardisation values “FVNormMin” and “FVNormMax”. Both of these values are also inputs at the FB.

If no input value is available, a value can be simulated at the FB via the “VSim” input. The simulation is activated via the “VSimEn” input bit (= TRUE). Output value “V” is simulated here directly. This means that the simulated value is not scaled. It's only checked to determine if the simulated values fall short of the “VNormMin” value or exceed the “VNormMax” value. If the limit values are not complied with, the corresponding limit value is read out as “V”.

22.5.3.3 Output value “V”

Input value “FV” is converted to output value “V” with the help of the standardisation values (“FVNormMin”, “FVNormMax”, “VNormMin” and “VNormMax”) and is read out from the FB. As is also the case for the standardisation values for the input value, the standardisation values for “V” are available as inputs at the FB. If “V” exceeds or falls short of its standardisation values (“VNormMin” and “VNormMax”), the corresponding standardisation value is read out as an output.

If the simulation is used, “Simulated Value” is set as the WQC. The simulated value is not scaled for read-out at output “V”. This means that only limit value monitoring takes place, but no conversion.

Output value (“V”) is frozen if the associated service is in the completed, stopped or aborted state. If the service is in the offline state, the value is set to 0.

22.5.3.4 Conversion of input value “FV” to output value “V”

The standardisation values, which can be read into the FB as inputs, are required for the input as well as for the output in order to convert the input value to an output value.

The conversion formula is as follows:

$$V = \frac{VNormMax - VNormMin}{FVNormMax - FVNormMin} * (FV - FVNormMin) + VNormMin$$

Input: FVNormMin and FVNormMax

Output: VNormMin and VNormMax

22.5.3.5 Unit of measure for “V”

A unit of measure can be entered for the output value via the “VUnit” input variable at the FB. The “UnitSettings” enumeration is used in order to specify a unit of measure. The unit of measure is only used for displays and has no influence on the conversion.

22.5.3.6 Monitoring the standardisation values

As described above, the standardisation values limit the input and output values and are used for conversion. It must also be assured that the minimum standardisation values are not greater than the associated maximum standardisation values. Otherwise, there’s no guarantee that the conversion and the graphics in the visualisation will include sensible values.

Furthermore, the difference between “FVNormMax” and “FVNormMin” must not be zero, because it’s not possible to divide by zero. In this case zero is read out at output value “V”.

22.5.3.7 Scaling values for the visualisation

One Min and one Max variable (VScIMin and VScIMax) are used to set the display limits of each visual representation of the output value. These variables are used exclusively for the representation of graphic elements in the visualisation. If output “FullScIEn” is set to TRUE, standardisation values “VNormMin” and “VNormMax” are used automatically as display limits. If the display limits are set incorrectly, they’re set automatically to the standardisation values. Refer to the section entitled “FB-specific worst quality code” in this regard.

22.5.4 FB-specific worst quality code (WQC)

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
96d	16#60	0	Simulation Value	VSimEn (= TRUE)
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
				VScIMax < VScImin ⇒ VScIMax = VNormMax ⇒ VScImin = VNormMin
				VScIMax < VNormMin or VScIMax > VNormMax ⇒ VScIMax = VNormMax
				VScImin < VNormMin or VScImin > VNormMax ⇒ VScImin = VNormMin
				VScIMax – VScImin = 0 ⇒ VScIMax = VNormMax ⇒ VScImin = VNormMin
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	At the beginning of each cycle

Tab. 143 DIntReportValue specific worst quality code

22.5.5 Persistent variables

Persistent variables are permanently stored in the controller so that they retain their current values even after a:

- Cold start
- Program download
- Online change

The variables are only restored to their initial values after a full reset (origin).

In order to assure that the controller persistently stores the variables, an additional object is added to the structure tree in CODESYS via **Application -> Add Object -> Persistent Variables**.

All of the instance paths of the persistent variables can be automatically added in this object.

The “persistent variables” object must be opened to this end. The **Declaration** tab appears in the top menu bar. **Add all instance paths** can be selected from this tab. The corresponding links are created as a result. If an error occurs, the “FestoPAToolkit” namespace must be added manually.

Note:

- It's even possible to make a selection specifying which variables will be persistently stored.
- Storage capacity for persistent variables at each controller must be taken into consideration. This is described in the user manuals of the respective controllers.

The following table lists the persistent variables for this FB. In order to assure trouble-free operation, these should also be used as persistent variables in the application.

Variable	Type	Default	Description
VNormMax	DINT	100	High Scale Value of Output Value (High limit of Output Value and scaling factor for V calculation)
VNormMin	DINT	0	Low Scale Value of Output Value (Low limit of Output Value and scaling factor for V calculation)

Tab. 144 DIntReportValue persistent variables

23 Character string service parameters

23.1 StringConfigParam

23.1.1 Brief description

The StringConfigParam function block (FB) can be used to define a string-based configuration parameter (character string) for a service. The parameter's current value is available at the VOut output. As opposed to a simple value specification, For example, when using a VAR_INPUT of the STRING type, this provides for the following extensions:

- Analogous to a service function block, this function block also has its own state (Offline/Operator/Automatic) and permits value specifications from various sources (Operator/Internal/External).
- VOut is only refreshed when the service allocated to the parameter switches from the offline to the operator or automatic state – otherwise, VOut remains constant. And thus a configuration parameter is above all suitable for specifying values which are required for the basic configuration of a service, and which are not permitted to change during the use of a service (For example, selection of the valve to be used for a dosing service).

23.1.2 Basic functions

23.1.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

23.1.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 145 Worst quality code enumeration

23.1.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

23.1.3 Functions

23.1.3.1 Service-parameter assignment

Fundamentally, there's an unequivocal service-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. When invoked, the parameter FB must also be informed of the associated service via the Service input.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
    Parameter1: StringConfigParam := (TagName := 'Parameter1', TagDescription := '');
END_VAR

(* Implementation example ST *)
SUPER^();
Parameter1(Service := THIS^);
```

23.1.3.2 Input value (Operator/Internal/External)

Due to the fact that there are three input values, a decision must be made as to which value is active (VOp, VInt or VExt), and the value must be further processed for the calculation of output VOut. The source mode is available to this and, which controls both inputs. Upon initialisation, the manual source is active, i.e. VOp. The mode of operation and handling of the source mode is explained in a separate section. VOp and VExt are not inputs at the FB, and can only be set as an OPC UA variable.

23.1.3.3 Output value “VOut”

The selected input value is read out as “VOut” as an output at the FB. VOut is only refreshed when the service allocated to the parameter switches from the offline to the operator or automatic state – otherwise, VOut remains constant.

23.1.3.4 Output “VRbk”

The “VRbk” output indicates the currently active input value, but without limiting the value to the standardisation values. “VRbk” is not updated when the parameter block is in the “Offline” state.

23.1.3.5 Output “VReq”

The “VRbk” output indicates the currently active input value, limited to the standardisation values. “VReq” is not updated when the parameter block is in the “Offline” state.

23.1.4 Source mode

23.1.4.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (external). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (VInt) the “SrcIntAct” output is set to TRUE. When the manual source is active (VExt), the “SrcExtAct” output bit is set to TRUE.

The internal source is active as a default setting.

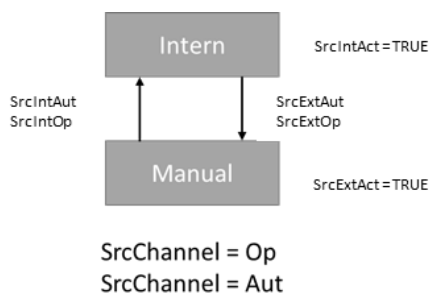


Fig. 155 Statemachine source mode

23.1.4.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutoP enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcExtAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAut” and “SrcExtAut” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcExtAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

23.1.4.3 Module-external access (OPC UA)

The “SrcIntOp” and “SrcExtOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “SrcIntOp” (internal source) or “SrcExtOp” (external source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcExtOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntOp” and “SrcExtOp”) are set simultaneously, “SrcIntOp” has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “SrcIntOp”, “SrcExtOp”, “SrcIntAut” or “SrcExtAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

23.1.5 Operation mode

23.1.5.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode.

The “StateChannel” input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the “StateOffAct” output is set to TRUE. When the operator mode is active “StateOpAct” is set to TRUE, and when the automatic mode is active “StateAutAct” is set to TRUE.

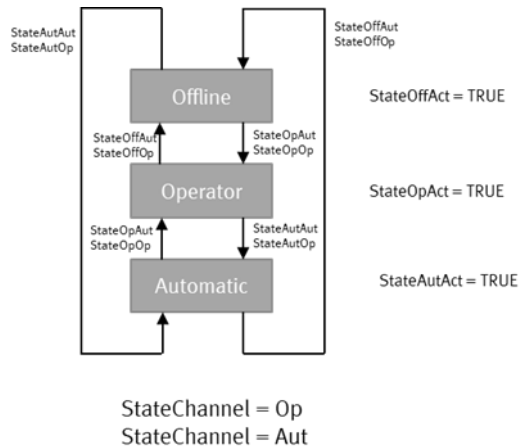


Fig. 156 Statemachine operation mode

23.1.5.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- “StateChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “StateOffAut” input (switch to offline mode), the “StateOpAut” input (switch to operator mode) or the “StateAutAut” input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the source is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

23.1.5.3 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.

- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

23.1.5.4 Synchronisation of the operation and source modes

As described above, the parameter block also has its own state (Offline/Operator/Automatic) and permits value specifications from various sources (Operator/Internal/External). Due to the fact that the parameter is unequivocally assigned to a certain service, in many cases it makes good sense to synchronise the state of the service and the parameter, as well as their channel for value specification. The Sync input can be used to this end. If this input is set to True (default value), the parameter’s own operation and source modes (see below) are ignored, and the current operation and source modes of the assigned service are used instead.

Note:

If the assigned service enters the offline state while synchronisation is activated, the parameter remains in its most recent state. If the parameter should also be set to the offline state, this transition must be triggered separately.

23.2 StringProcedureParam

23.2.1 Brief description

The StringProcedureParam function block (FB) can be used to define a string-based procedure parameter (character string) for a service procedure. The parameter's current value is available at the VOut output. As opposed to a simple value specification, For example, when using a VAR_INPUT of the STRING type, this provides for the following extensions:

- Analogous to a service function block, this function block also has its own state (Offline/Operator/Automatic) and permits value specifications from various sources (Operator/Internal/External).
- VOut is only refreshed when the service allocated to the parameter is started with the procedure assigned to the parameter (i.e. its Starting state is executed) – otherwise, VOut remains constant. And thus a procedure parameter is above all suitable for specifying values which are required for a certain procedure of a service, and which are not permitted to change during execution of the service (For example, specification of a duration for a time-limited dosing procedure).
-

23.2.2 Basic functions

23.2.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

23.2.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 146 Worst quality code enumeration

23.2.2.3 OSLevel

The OSLevel variable provides a simple mechanism for monitoring the writing of multiple operating levels. A corresponding number is assigned to each operating level (e.g. local/central control station). If the number of the operating level and the number of this variable match, the operating device's controls should permit the write command. Otherwise, the entry fields should be inactive. This prevents simultaneous write access from several operating levels.

The variable doesn't trigger any functionality in the FB and is not checked. It only serves as a persistent storage location so that all operating stations can access one and the same value. The variable is not available as an input at the FB – it can only be written via OPC UA.

23.2.3 Functions

23.2.3.1 Service-procedure-parameter assignment

Fundamentally, there's an unequivocal service-procedure-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. Furthermore, the procedure to which the parameter is assigned must be defined via the ProcedureID parameter during initialisation. When invoked, the parameter FB must also be informed of the associated service via the Service input.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
    Timelimited: ServiceProcedure := (TagName := 'time-limited', TagDescription := '', ProcedureID := 1,
    IsSelfCompleting := TRUE);
    Time: StringProcedureParam := (TagName := 'Time', TagDescription := '', ProcedureID := 1);
END_VAR

(* Implementation example ST *)
SUPER^();
Timelimited(Service := THIS^);
Time(Service := THIS^);
```

23.2.3.2 Input value (Operator/Internal/External)

Due to the fact that there are three input values, a decision must be made as to which value is active (VOp, VInt or VExt), and the value must be further processed for the calculation of output VOut. The source mode is available to this and, which controls both inputs. Upon initialisation, the manual source is active, i.e. VOp. The mode of operation and handling of the source mode is explained in a separate section. VOp and VExt are not inputs at the FB, and can only be set as an OPC UA variable.

23.2.3.3 Output value “VOut”

The selected input value is read out as “VOut” as an output at the FB. VOut is only refreshed when the service allocated to the parameter is started with the procedure assigned to the parameter (i.e. its Starting state is executed) – otherwise, VOut remains constant.

23.2.3.4 Output “VRbk”

The “VRbk” output indicates the currently active input value, but without limiting the value to the standardisation values. “VRbk” is not updated when the parameter block is in the “Offline” state.

23.2.3.5 Output “VReq”

The “VRbk” output indicates the currently active input value, limited to the standardisation values. “VReq” is not updated when the parameter block is in the “Offline” state.

23.2.4 Source mode

23.2.4.1 Brief description

The source mode is a state machine for selecting a source, in the event that a value can be influenced by two different sources.

Differentiation is made between internal and manual value specification. The “SrcChannel” input at the FB is used to specify whether the FB is permitted to switch the source mode directly (internal), or if an operator is permitted to make changes via a faceplate (external). Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the SrcChannel variable.

When the internal source is active (VInt) the “SrcIntAct” output is set to TRUE. When the manual source is active (VExt), the “SrcExtAct” output bit is set to TRUE.

The internal source is active as a default setting.

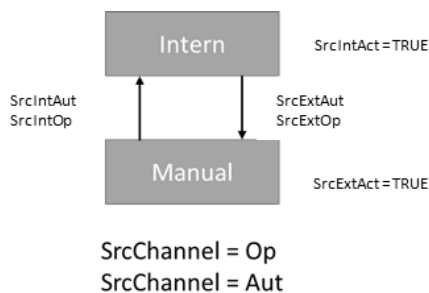


Fig. 157 Statemachine source mode

23.2.4.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the source mode directly at the FB:

- “SrcChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “SrcIntAut” input (switch from manual to internal) or “SrcExtAut” input (switch from manual to internal) to TRUE.
- If the request is acknowledged by the FB, “SrcIntAut” and “SrcExtAut” are set to FALSE.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntAut” and “SrcExtAut”) are set simultaneously, “SrcIntAut” has higher priority and the source is changed to or remains at internal.

23.2.4.3 Module-external access (OPC UA)

The “SrcIntOp” and “SrcExtOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a source change can be requested via OPC UA:

- “SrcChannel” must be set to “OP” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “SrcIntOp” (internal source) or “SrcExtOp” (external source) to TRUE.
- After the request, the FB sets both variables (“SrcIntOp” and “SrcExtOp”) to FALSE – i.e. the request has been acknowledged by the FB.
- The currently selected source can be read out via the “SrcIntAct” and “SrcExtAct” outputs. Depending on which source is active, the corresponding output is set to TRUE.

If the FB sets the inputs back to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “SrcChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If both inputs (“SrcIntOp” and “SrcExtOp”) are set simultaneously, “SrcIntOp” has higher priority and the source is changed to or remains at internal.

Note:

The source mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “SrcIntOp”, “SrcExtOp”, “SrcIntAut” or “SrcExtAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it’s important to ensure that these inputs are not permanently assigned to the FB.

23.2.5 Operation mode

23.2.5.1 Brief description

The operation mode is a state machine with three states, which determines whether the FB is in the offline, operator or automatic mode.

The “StateChannel” input at the FB is used to specify whether the FB is permitted to switch the operation mode directly, or if an operator is permitted to make changes via a faceplate. Depending on this variable, either the operator signals (*Op variables) or the internal input variables at the FB (*Aut variables) have precedence. The point in time at which switching is permitted can be defined by the FB with the help of the StateChannel variable. The FB is always in the offline state upon initialisation.

When the offline mode is active the “StateOffAct” output is set to TRUE. When the operator mode is active “StateOpAct” is set to TRUE, and when the automatic mode is active “StateAutAct” is set to TRUE.

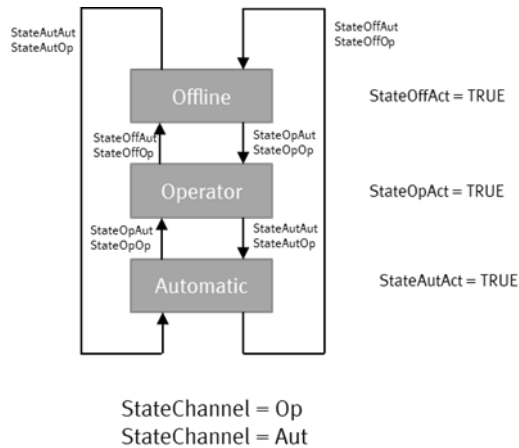


Fig. 158 Statemachine operation mode

23.2.5.2 Module-internal access (directly at the FB)

The following procedure must be followed in order to change the operation mode directly at the FB:

- “StateChannel” must be set to “Aut” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set the “StateOffAut” input (switch to offline mode), the “StateOpAut” input (switch to operator mode) or the “StateAutAut” input (switch to automatic mode) to TRUE.
- If the request is acknowledged by the FB, the inputs are set back to FALSE.
- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn’t necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they’re acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffAut”, “StateOpAut”, “StateAutAut”) are set simultaneously, the offline mode has higher priority and the source is changed to or remains in the offline mode. Since it’s not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

23.2.5.3 Module-external access (OPC UA)

The “StateOffOp”, “StateOpOp” and “StateAutOp” variables are used for access via OPC UA.

The following procedure must be followed in order to ensure that a state change can be requested via OPC UA:

- “StateChannel” must be set to “Op” via the ChannelAutOp enumeration (can only be changed at the FB or via the PA Toolkit’s faceplate).
- Set “StateOffOp” (offline), “StateOpOp” (operator) or “StateAutOp” (automatic) to TRUE.
- After the request, all three variables (“StateOffOp”, “StateOpOp” and “StateAutOp”) must be set to FALSE by the FB – i.e. the request has been acknowledged by the FB.

- Which mode is active can be read out via the “StateOffAct”, “StateOpAct” or “StateAutAct” output. The output bit of the active mode is set to TRUE.

If the FB sets the inputs to FALSE, this doesn't necessarily mean that the command will be executed – it only means that the command has been received by the FB. For example, if “StateChannel” is set to Op variables, requests are not processed by the FB although they're acknowledged. Invalid requests are not saved for a later point in time.

If two or more inputs (“StateOffOp”, “StateOpOp”, “StateAutOp”) are set simultaneously, the offline mode has higher priority and the state is changed to or remains in the offline mode. Since it's not possible to switch directly from the automatic mode to the offline mode, the operator mode has a higher priority than the automatic mode in this case.

Note:

The operation mode works on the basis of a handshake procedure. If a control command is sent to the FB via variable “StateOffOp”, “StateOpOp”, “StateAutOp”, “StateOffAut”, “StateOpAut” or “StateAutAut”, the variable is set back to FALSE in order to acknowledge the request. For this reason, it's important to ensure that these inputs are not permanently assigned to the FB.

23.2.5.4 Synchronisation of the operation and source modes

As described above, the parameter block also has its own state (Offline/Operator/Automatic) and permits value specifications from various sources (Operator/Internal/External). Due to the fact that the parameter is unequivocally assigned to a certain service, in many cases it makes good sense to synchronise the state of the service and the parameter, as well as their channel for value specification. The Sync input can be used to this end. If this input is set to True (default value), the parameter's own operation and source modes (see below) are ignored, and the current operation and source modes of the assigned service are used instead.

Note:

If the assigned service enters the offline state while synchronisation is activated, the parameter remains in its most recent state. If the parameter should also be set to the offline state, this transition must be triggered separately.

23.3 StringProcessValueOut

23.3.1 Brief description

The StringProcessValueOut function block (FB) is used for continuous feedback of a string-based value (character string) from a service procedure.

23.3.1.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

23.3.1.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst Quality Code Enumeration			
Byte	Byte	Priorität	Beschreibung
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 147 Worst quality code enumeration

23.3.2 Functions

23.3.2.1 Service-procedure-parameter assignment

Fundamentally, there's an unequivocal service-procedure-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. Furthermore, the procedure to which the parameter is assigned must be defined via the ProcedureID parameter during initialisation.

```
(* Declaration example *)
FUNCTION_BLOCK Stir EXTENDS ServiceControl
VAR
    Timelimited: ServiceProcedure := (TagName := 'time-limited', TagDescription := '', ProcedureID := 1,
    IsSelfCompleting := TRUE);
    Level: StringProcessValueOut := (TagName := 'Level', TagDescription := '', ProcedureID := 1);
END_VAR

(* Implementation example ST *)
SUPER^();
Timelimited(Service := THIS^);
Level(Service := THIS^);
```

23.3.2.2 Input value “Text”

A character string can be entered via the “Text” input variable. The character string can be entered directly or via a variable.

23.4 StringReportValue

23.4.1 Brief description

The StringReportValue function block (FB) is used to make string-based value (character string) available from a service procedure for the purpose of documentation or verification. The FB extends the functionality of a StringProcessValueOut such that the value of the parameter ("V") is frozen if the associated service is in the completed, stopped or aborted state. This makes it possible for an interconnected documentation system to read out the parameter's value in these states, e.g. for the purpose of verification.

23.4.2 Basic functions

23.4.2.1 Identification

At least one symbol is available for the visualisation of each function block. In order to ensure that the symbol can be matched up with the hardware unit in the visualisation, a tag name and a tag description can be entered when setting up the symbol. Up to 16 characters can be used for the tag name and up to 32 characters for the tag description – additionally entered characters are omitted. Due to the fact that these values are also relevant for MTP export, in which they're statically processed, both values must be entered in the following format: "Tagname".

23.4.2.2 Worst quality code (WQC)

The worst quality code indicates the current status of the function block and its signals. The worst status always has highest priority and is read out via the WQC output. All descriptions are listed in the following table. Specific error handling for this FB is explained in the section entitled "FB-specific worst quality code".

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 148 Worst quality code enumeration

23.4.3 Functions

23.4.3.1 Service-procedure-parameter assignment

Fundamentally, there's an unequivocal service-procedure-parameter assignment. In order to make this assignment plainly apparent, a parameter must always be declared and initialised in the declaration section of the corresponding service FB. Furthermore, the procedure to which the parameter is assigned must be defined via the ProcedureID parameter during initialisation. When invoked, the parameter FB must also be informed of the associated service via the Service input.

```
(* Declaration example *)  
FUNCTION_BLOCK Stir EXTENDS ServiceControl  
VAR  
    Timelimited: ServiceProcedure := (TagName := 'time-limited', TagDescription := '', ProcedureID := 1,  
    IsSelfCompleting := TRUE);  
    Level: StringReportValue := (TagName := 'Level', TagDescription := '', ProcedureID := 1);  
END_VAR  
  
(* Implementation example ST *)  
SUPER^();  
Timelimited(Service := THIS^);  
Level(Service := THIS^);
```

23.4.3.2 Input value “TextIn”

A character string can be entered via the “TextIn” input variable. The character string can be entered directly or via a variable.

24 Parameter blocks

Three parameter blocks are provided in order to be able to display and set service parameters in the visualisation. There's one each for REAL, STRING and BOOL parameters. Each FB has inputs to which the service parameters can be linked. The corresponding parameter symbol for the service must be added in the visualisation. During runtime, the faceplate with the linked parameters can be opened by clicking the parameter symbol. Parameters can be read and written here. The FBs don't have any further functions – they're only used for display in the visualisation.

24.1 Symbols

24.1.1.1 hBoolParameter



Fig. 159 Symbol hBoolParameter

24.1.1.2 hRealParameter



Fig. 160 Symbol hRealParameter

24.1.1.3 hStringParameter



Fig. 161 Symbol hStringParameter

24.2 Faceplates Bool-Parameter

24.2.1.1 Param 1-4

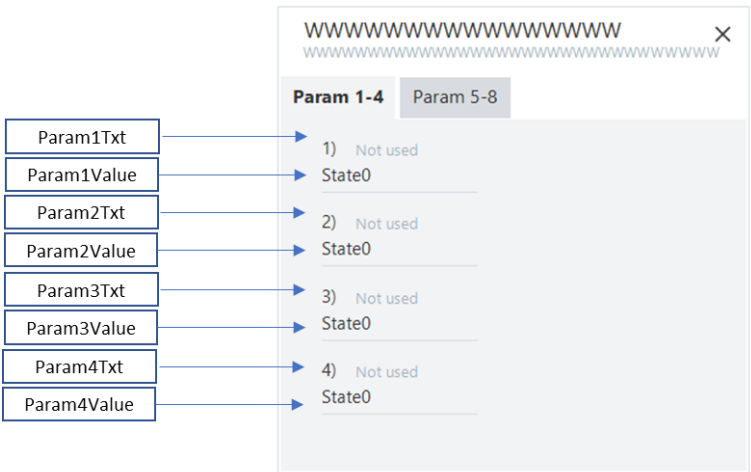


Fig. 162 Bool-Parameter Param 1-4 faceplate

24.2.1.2 Param 5-8

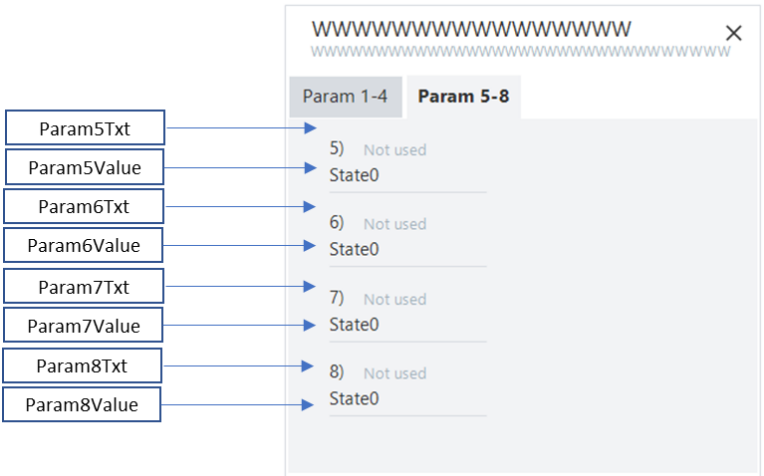


Fig. 163 Bool-Parameter Param 5-8 faceplate

24.3 Faceplates Real-Parameter

24.3.1.1 Param 1-4

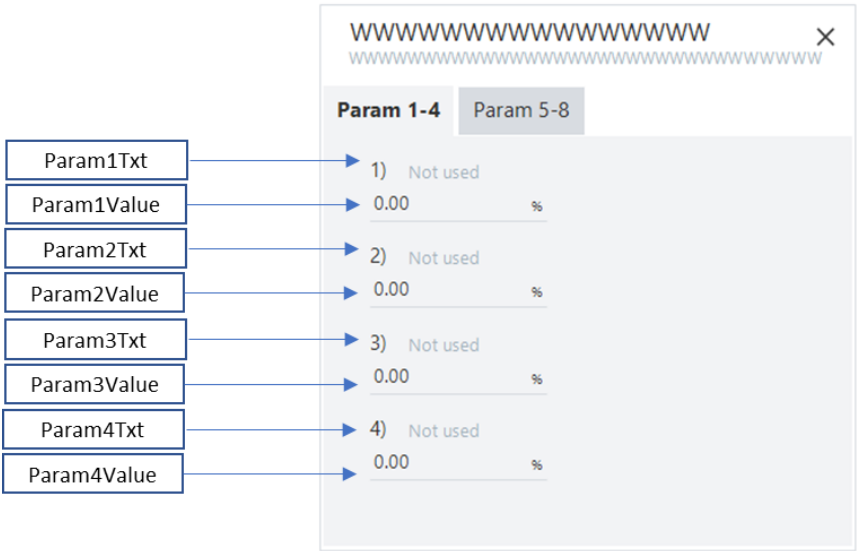


Fig. 164 Real-Parameter Param 1-4 faceplate

24.3.1.2 Param 5-8

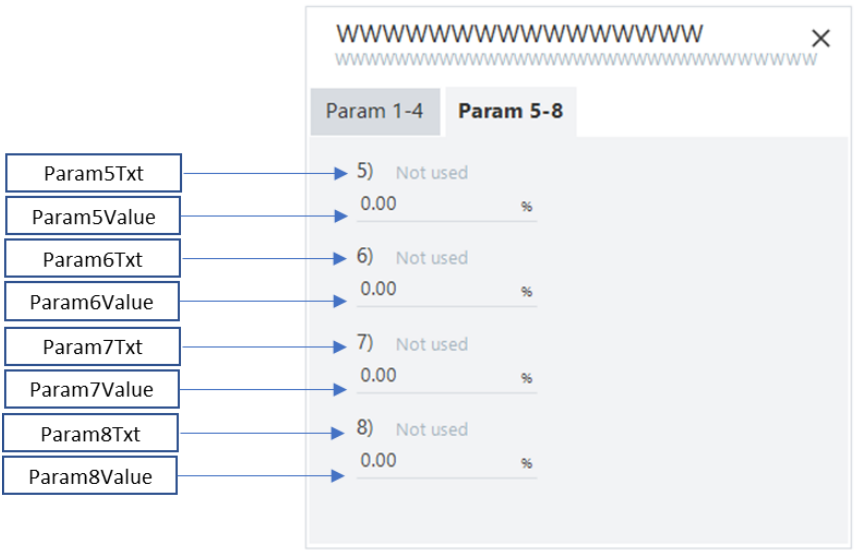


Fig. 165 Real-Parameter Param 5-8 faceplate

24.4 Faceplates String-Parameter

24.4.1.1 Param 1-4

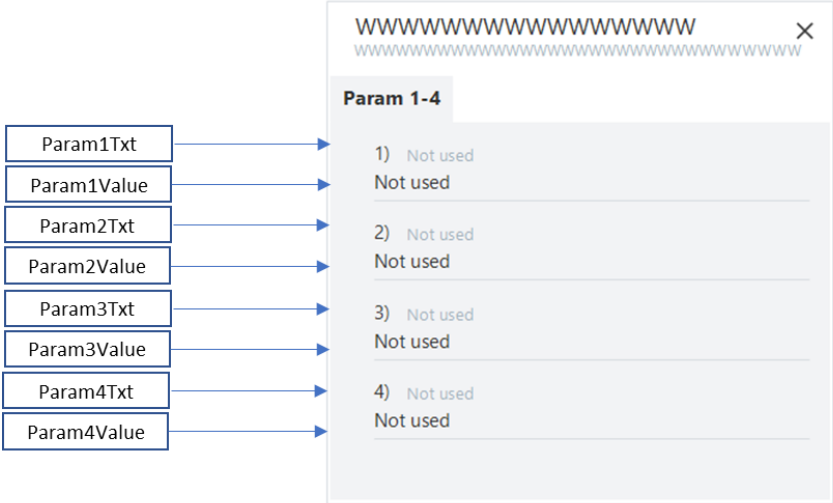


Fig. 166 String-Parameter Param 1-4 faceplate

25 Help functions and enumeration

25.1 Help functions

25.1.1 Scaler

The scaler function block can be used for conversion from an input value “FV” to an output value “V”. The scaling values from “FV” (“FVNormMin” and “FVNormMax”) and the outputs (“VNormMin” and “VNormMax”) are additionally transferred as well. A WQC (worst quality code) is returned as an output value, as well as the converted “V” value as REAL and as UINT.

The following WQCs can occur with this FB:

FB-specific worst quality code				
Byte	Byte	Priority	Description	Reason/error
40d	16#28	2	Bad, conditioned by process	FVNormMin > FVNormMax
				VNormMin > VNormMax
				FVNormMax-FVNormMin = 0 ⇒ V = 0
				V < VNormMin ⇒ V = VNormMin
				V > VNormMax ⇒ V = VNormMax
128d	16#80	6	Good	No error
255d	16#FF	7	No QC available	Default value

Tab. 149 Scaler specific worst quality code

25.2 Enumeration

25.2.1 WorstQualityCode enumeration

WorstQualityCode is an enumeration and represents the various status displays which can be assumed by the worst quality code.

Worst quality code enumeration			
Byte	Byte	Priority	Description
96d	16#60	0	Simulation Value
00d	16#00	1	Bad, conditioned by device
40d	16#28	2	Bad, conditioned by process
104d	16#68	3	Uncertain, conditioned by device
120d	16#78	4	Uncertain, conditioned by process
164d	16#A4	5	Maintenance Request
128d	16#80	6	Good
255d	16#FF	7	No QC available (default)

Tab. 150 Worst quality code enumeration

25.2.2 UnitSettings enumeration

All units are included in the enumeration which can be used in the PA Toolkit.

25.2.3 ChannelAutoOp enumeration

This enumeration is used for switching the Op and Aut variables at an FB. Each FB which uses a source mode has an “SrcChannel” input. If the operation mode is used, the FB has a “StateChannel” input. This input can be used to specify whether the OP variables or the Aut variables will be used for control. The enumeration consists of the “Aut” and “Op” entries and can be used for precisely these two inputs at the FB.

25.2.4 LockLogic enumeration

The basic logic operation can be set for the lock display. This enumeration provides logic options AND and OR.

25.2.5 Command enumeration

Command is an enumeration which represents the various possible commands for triggering state changes for the state machine of a service function block (ServiceControl, SimpleContinuousServiceControl and SimpleSelfCompletingServiceControl).

The values are coded in accordance with the following table:

Value (decimal)	Value (hexadecimal)	Name
0	16#0	NoCommand
2	16#2	Reset
4	16#4	Start
8	16#8	Stop
16	16#10	Hold
32	16#20	Unhold
64	16#40	Pause
128	16#80	Resume
256	16#100	Abort
512	16#200	Restart
1024	16#400	Complete

Tab. 151 Command enumeration

25.2.6 SMState enumeration

SMState is an enumeration which represents the various possible states for the state machine of a service function block (ServiceControl, SimpleContinuousServiceControl and SimpleSelfCompletingServiceControl).

The values are coded in accordance with the following table:

Value (decimal)	Value (hexadecimal)	Name
4	16#4	Stopped
8	16#8	Starting
16	16#10	Idle
32	16#20	Paused
64	16#40	Execute
128	16#80	Stopping
256	16#100	Aborting
512	16#200	Aborted
1024	16#400	Holding
2048	16#800	Held
4096	16#1000	Unholding
8192	16#2000	Pausing
16384	16#4000	Resuming
32768	16#8000	Resetting
65536	16#10000	Completing
131072	16#20000	Completed

Tab. 152 SMState enumeration